ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОУВПО «Пермский государственный университет»

Л.Н. Лядова, В.В. Ланин

Microsoft Office: от начинающего пользователя до профессионала

Учебно-методическое пособие
В 2 частях

Часть 2. Основы офисного программирования

Пермь 2007

УДК 681.3
ББК 32.97
Л97

Лядова Л.Н.

Л97 Microsoft Office: от начинающего пользователя до профессионала: В 2 ч.: учеб.-метод. пособие / Л.Н. Лядова, В.В. Ланин; Перм. ун-т. – Пермь, 2007. – Ч. 2: Основы офисного программирования – 388 с.: ил.

ISBN 5-7944-0835-9

Рассматриваются общие вопросы разработки офисных приложений на основе пакета Microsoft Office, излагаются основы языка Visual Basic for Applications (VBA). Описываются правила записи программ на VBA, средства создания и подключения макросов и возможности редактора Visual Basic для разработки макросов, пользовательского интерфейса приложений. Включены сведения об организации доступа к базам данных из приложений MS Office, об интеграции различных приложений пакета. Приведены примеры создания приложений на основе интегрированного офисного пакета. Описан пакет Office CASE, разработанный на основе MS Office, позволяющий создавать и обрабатывать системы взаимосвязанных документов.

Пособие предназначено для преподавателей, ведущих занятия по языкам программирования, а также по дисциплинам, связанным с изучением электронного офиса, средств автоматизации делопроизводства и документооборота. Может быть полезно для пользователей, изучающих офисный пакет на углубленном уровне.

Рецензент – доктор физ.-мат. наук, профессор, директор учебного центра «Информатика» *С.В. Русаков*

Печатается в соответствии с решением редакционно-издательского совета Пермского университета

Данное пособие является победителем конкурса, проведенного Пермским государственным университетом в ходе реализации инновационной образовательной программы «Формирование информационно-коммуникационной компетентности выпускников классического университета в соответствии с потребностями информационного общества» в рамках приоритетного национального проекта «Образование».

УДК 681.3
ББК 32.97

ISBN 5-7944-0835-9

ОГЛАВЛЕНИЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА 10
Уровень предварительной подготовки слушателей 10
Образовательные цели и задачи 11
Знания и навыки, получаемые в ходе изучения курса 12
Содержание занятий 13
Вопросы для самостоятельного изучения 13
Содержание приложений 14
Итоговые работы 14

ВВЕДЕНИЕ 15

ГЛАВА 1. ОБЩАЯ ХАРАКТЕРИСТИКА ВОЗМОЖНОСТЕЙ VBA 16

ГЛАВА 2. ОСНОВНЫЕ ПРИНЦИПЫ РАЗРАБОТКИ ПРИЛОЖЕНИЙ 18
Вопросы для самопроверки 19

ГЛАВА 3. ОСНОВНЫЕ ПРАВИЛА РАЗРАБОТКИ ИНТЕРФЕЙСА 20
Вопросы для самопроверки 22

ГЛАВА 4. VISUAL BASIC ДЛЯ ПРИЛОЖЕНИЙ 23
Вопросы для самопроверки 24

ГЛАВА 5. ОСНОВЫ АЛГОРИТМИЗАЦИИ 25
5.1. Определение и свойства алгоритмов 25
5.2. Основные этапы и методы разработки алгоритмов 28
5.3. Основные способы описания алгоритмов 30
5.4. Основные управляющие структуры, применяемые для записи алгоритмов 31
5.5. Создание программы 38
Вопросы для самопроверки 39

ГЛАВА 6. ВВЕДЕНИЕ В ЯЗЫК И РЕДАКТОР VISUAL BASIC 40
6.1. Окно Редактора Visual Basic 40
6.2. Основные правила записи программ на VBA 43
6.2.1. Описания в программах на языке VBA 45
6.2.2. Структуризация программ VBA и правила записи кода процедур 47
6.2.3. Использование параметров 48
6.2.4. Типы данных VBA 51
6.2.5. Описания переменных 51
6.2.6. Объекты в программах на VBA и объектные переменные 55
6.2.7. Создание объектов пользователя 58
6.2.8. Описание и использование констант 62
6.2.9. Массивы в VBA 64
6.2.10. Описание типов данных пользователя 66
6.2.11. Запись выражений на VBA 67
6.2.12. Рекурсивные вычисления 69
6.2.13. Процедуры-подпрограммы 69
6.2.14. Процедуры-функции 73
6.2.15. Процедуры-свойства 75
6.2.16. Создание процедур 76
6.2.17. Управляющие структуры VBA и события 76
6.3. Запуск программ 83
6.4. Отладка программ 85
6.5. Обработка ошибок 85
6.6. Преобразования типов 91
6.7. Функции для работы с датами и временем 92
6.8. Управление файлами 94
6.9. Функции проверки 95
6.10. Функции работы со строками 96
6.11. Взаимодействие с пользователем 97
6.12. Пользовательские формы и диалоговые окна 97
6.13. Создание пользовательских меню и панелей инструментов 111
6.14. Разработка контекстного меню 118
6.15. Открытие проектов, экспорт и импорт 121
Вопросы для самопроверки 123
Запуск программы и работа в редакторе VBA 123
Структура программы, понятие подпрограммы и функции 123
Типы данных и описания в VBA 123
Выражения в VBA 124

Управляющие структуры и события в программах на VBA ... *125*
Проектирование и разработка интерфейса ... *126*
Проекты: открытие, экспорт и импорт ... *126*
Обработка ошибок ... *126*

ГЛАВА 7. ИСПОЛЬЗОВАНИЕ WIN32 API ... 127
Вопросы для самопроверки ... 131

ГЛАВА 8. ИСПОЛЬЗОВАНИЕ ШАБЛОНОВ И НАДСТРОЕК ... 132
Вопросы для самопроверки ... 135

ГЛАВА 9. РАЗРАБОТКА ПРИЛОЖЕНИЙ С ПОМОЩЬЮ EXCEL 136
9.1. Иерархия объектов Excel ... 136
9.2. Обзор основных объектов Excel ... 142
9.2.1. Объект Application ... *143*
9.2.2. Объект Workbook ... *147*
9.2.3. Объект Worksheet ... *150*
9.2.4. Объект Range ... *151*
9.2.5. Графические объекты ... *154*
9.2.6. Элементы управления и обработка событий ... *155*
9.3. Представление данных и вычисления в таблицах Excel ... 159
9.3.1. Ссылки на ячейки рабочих листов Excel ... *160*
9.3.2. Работа с формулами ... *164*
9.3.3. Использование функций ... *167*
9.3.4. Создание пользовательских функций рабочего листа ... *168*
9.3.5. Массивы Excel ... *170*
9.3.6. Определение связей между таблицами ... *172*
9.3.7. Консолидация данных ... *175*
9.3.8. Форматирование данных в таблицах ... *181*
9.4. Анализ данных и подведение итогов ... 191
9.4.1. Создание сводных таблиц ... *192*
9.4.2. Подведение промежуточных итогов ... *208*
9.4.3. Определение частичных сумм ... *209*
9.4.4. Создание диаграмм ... *211*
9.5. Работа со структурой данных ... 212
9.6. Подбор параметров и поиск решения ... 214

9.6.1. Подбор параметра ... *215*
9.6.2. Поиск решения ... *216*
9.6.3. Использование сценариев ... *217*
9.7. Поиск и отбор данных в таблицах Excel ... 219
9.7.1. Использование автофильтра ... *219*
9.7.2. Работа с расширенным фильтром ... *221*
9.7.3. Поиск данных по подписям строк и столбцов ... *223*
9.8. Упорядочение данных в таблицах ... 224
9.9. Защита приложений Excel ... 224
9.9.1. Защита от ошибок при вводе данных ... *225*
9.9.2. Защита данных от несанкционированного доступа ... *226*
Вопросы для самопроверки ... 228
Иерархия объектов Excel ... *228*
Представление данных и вычисления в таблицах Excel ... *229*
Анализ данных и подведение итогов ... *230*
Работа со структурой данных ... *230*
Подбор параметров и поиск решения ... *230*
Поиск и отбор данных в таблицах Excel ... *231*
Упорядочение данных в таблицах ... *231*
Защита приложений excel ... *231*

ГЛАВА 10. РАЗРАБОТКА ПРИЛОЖЕНИЙ С ПОМОЩЬЮ WORD . 232
10.1. Модель объектов Word ... 233
10.1.1. Объект Document ... *233*
10.1.2. Другие дочерние объекты приложения Word ... *239*
10.2. Работа с полями ... 241
10.3. Переменные документа ... 245
10.4. Использование формул в документах Word ... 246
10.5. Использование элементов управления в документах Word ... 251
10.6. Создание шаблонов документов на основе таблиц ... 252
10.7. Защита документов Word и кода ... 256
Вопросы для самопроверки ... 260

ГЛАВА 11. ИСПОЛЬЗОВАНИЕ AUTOMATION ПРИ ИНТЕГРИРОВАНИИ КОМПОНЕНТОВ MICROSOFT OFFICE ... 262
11.1. Технология ActiveX ... 262
11.2. Компонентная модель объектов, внедрение и связывание ... 263
11.3. Создание объектов Automation ... 267
11.4. Получение доступа к объекту ... 273
11.5. Пример вычислений в Access с помощью Excel ... 276
11.6. Управление связанными и внедренными объектами с помощью программного кода ... 279
Вопросы для самопроверки ... 282

ГЛАВА 12. ОРГАНИЗАЦИЯ ДОСТУПА К ВНЕШНИМ ДАННЫМ ... 283
12.1. Доступ к данным из клиентских приложений ... 283
12.2. Интерфейс ODBC ... 284
12.3. Использование DAO для доступа к данным ... 288
12.3.1. Модель объектов DAO ... 289
12.3.2. Создание рабочей области и открытие источника данных ... 292
12.3.3. Исследование структуры таблиц БД ... 293
12.3.4. Операции над данными ... 294
12.4. Интеграция данных и работа в сети ... 296
12.4.1. Доступ к данным Excel из СУБД Access ... 296
12.5. Выбор метода доступа к данным ... 297
Вопросы для самопроверки ... 298

ГЛАВА 13. ОСОБЕННОСТИ РАННЕГО И ПОЗДНЕГО СВЯЗЫВАНИЯ ... 300
13.1. Задача связывания ... 300
13.1.1. Раннее связывание ... 300
13.1.2. Позднее связывание ... 303
Вопросы для самопроверки ... 304

ГЛАВА 14. ОБЩИЕ РЕКОМЕНДАЦИИ ПО РАЗРАБОТКЕ ПРИЛОЖЕНИЙ ... 305
Вопросы для самопроверки ... 306

ГЛАВА 15. РАЗРАБОТКА СПРАВОЧНОЙ СИСТЕМЫ ... 307
15.1. Использование всплывающих подсказок ... 307
15.2. Контекстная справка «Что это такое?» ... 307

15.3. Подготовка файла справки ... 308
Вопросы для самопроверки ... 309

ГЛАВА 16. ДОПОЛНИТЕЛЬНЫЕ РЕКОМЕНДАЦИИ ... 309
16.1. Установка приложения ... 309
16.2. Повышение эффективности работы приложения ... 310
Вопросы для самопроверки ... 311

ПРИЛОЖЕНИЕ 1. АВТОМАТИЗИРОВАННОЕ РАБОЧЕЕ МЕСТО ОПЕРАТОРА АВТОЗАПРАВОЧНОЙ СТАНЦИИ ... 312
Постановка задачи ... 312
Разработка структуры рабочей книги ... 312
Настройка пользовательского интерфейса ... 315
Разработка экранных форм ... 318

ПРИЛОЖЕНИЕ 2. АВТОМАТИЗАЦИЯ ЗАПОЛНЕНИЯ БЛАНКА С ПОМОЩЬЮ ПРОГРАММЫ ТЕКСТОВОГО ПРОЦЕССОРА WORD ... 327
Постановка задачи ... 327
Разработка документа ... 327
Разработка экранной формы ... 331

ПРИЛОЖЕНИЕ 3. СИСТЕМА АВТОМАТИЗАЦИИ ПОДГОТОВКИ ДОКУМЕНТОВ НА ОСНОВЕ ПАКЕТА MICROSOFT OFFICE ... 334
Введение ... 334
Требования к системе Office CASE ... 337
Интеллектуальность документа ... 337
Однократный ввод данных и их совместное использование ... 338
Дружественный интерфейс – «ориентация на человека» ... 339
Мобильность ... 340
Независимость от версии Microsoft Office ... 341
Поддержка языка разметки документов XML ... 342
Общий сценарий использования ... 343
Структура Office CASE ... 346
Office CASE Document Framework ... 346
Структура скрытого раздела документа ... 348
Подсистемы программного проекта документа ... 352

Менеджер закладок 352
Менеджер классификаторов 352
Подсистема управления интерфейсом пользователя 354
Подсистема контроля данных 355
Подсистема информационного обмена 356
Office CASE Documents Studio *357*
Дизайнер системы документов 359
Редактор классификаторов 360
Редактор правил 360
Редактор закладок 363
Редактор блоков данных 364
ПРИМЕР ПРИМЕНЕНИЯ OFFICE CASE 367
Постановка задачи *367*
Реализация системы документов *369*
Документы «Подсистемы автоматизации подготовки отчетов» 369
Требования к квалификации пользователей 374
Установка приложения и подготовка к работе 374
Интерфейс первичных документов 377
Интерфейс сводных документов 382
СПИСОК ЛИТЕРАТУРЫ 386

Пояснительная записка

Данный курс ориентирован на слушателей, не являющихся специалистами в области информатики и программирования, но изучающих современные инфомационные технологии, применяющих их в своей повседневной деятельности.

Офисные технологии – наиболее широко распространенные информационные технологии, используемые абсолютным большинством пользователей персональных компьютеров в своей повседневной деятельности. Очень многие виды работ при этом являются рутинными, трудоемкими, связанными с повторением одних и тех же типовых операций, обработкой большого количества данных. Многие из этих операций могут быть автоматизированы средствами приложений Microsoft Office без привлечения разработчиков-программистов.

Программа курса включает вопросы, позволяющие слушателям получить начальные знания, умения и навыки в области разработки и использования приложений на основе пакета MS Office, достаточные для создания простейших приложений, позволяющих автоматизировать типовые трудоемкие операции, получить базовые навыки объектно-ориентированного программирования на основе иерархии объектов MS Office.

Уровень предварительной подготовки слушателей

Для успешного освоения предлагаемого для изучения в курсе материала специальная предварительная подготовка не требуется: изложение материала соответствует уровню пользователя, освоившего основные приемы работы с приложениями MS Office и имеющего базовые знания в области алгоритмизации, соответствующие уровню, на котором они изучаются в курсе информатики на специальностях, не связанных с подготовкой специалистов в области информационных технологий.

Образовательные цели и задачи

Основная цель курса – ознакомление слушателей с основами офисных технологий на уровне, обеспечивающем возможность применения полученных знаний и навыков в повседневной деятельности специалистов для автоматизации своих рабочих мест.

Специалисты, прослушавшие теоретический курс и выполнившие программу практикума, должны быть подготовлены к самостоятельной работе с приложениями пакета Microsoft Office для выполнения типовых операций, связанных с подготовкой документов, электронных презентаций и выполнением расчетов, на своих рабочих местах. Кроме того, слушатели должны научиться организовывать свои рабочие места, настраивать программы для эффективного использования компьютера в своей работе и создавать простейшие приложения, автоматизирующие типовые операции по созданию документов и обработке данных с помощью программ интегрированного пакета Microsoft Office.

Для достижения поставленной цели при изучении курса решаются следующие *задачи*:

- Формирование у слушателей общего представления о назначении и возможностях интегрированного пакета MS Office, его структуре, модели объектов (документов, их элементов), их свойствах и реализованных приложениями операциях.
- Знакомство с визуальной средой разработки офисных приложений – редактором Visual Basic for Applications (VBA).
- Получение слушателями базовых навыков создания простейших приложений на основе программ пакета Microsoft Office.

Встроенный язык пакета Visual Basic for Applications (VBA) предоставляет даже пользователю-непрограммисту богатые возможности по созданию собственных автоматизированных рабочих мест на основе автоматизации подготовки и обработки документов MS Office. Средства создания макросов позволяют сгенерировать приложения, автоматизирующие операции, часто выполняемые пользователями. Минимальные знания в области алгоритмизации и программировнаия позволяют в значительной мере расширить возможности автоматизации документов.

Основная задача преподавателя, ведущего занятия по курсу, – снять у слушателей психологический барьер, мешающий эффективному использованию возможностей пакета, обеспечивающих средства создания офисных приложений с минимальными затратами труда.

Знания и навыки, получаемые в ходе изучения курса

По окончании курса слушатели должны

- *знать*:
  - основные типы данных и управляющие конструкции VBA;
  - объектную модель приложений MS Office;
  - основные приемы разработки офисных приложений на основе MS Office;
  - методы доступа к данным из офисных приложений;
- *уметь*:
  - создавать документы приложений MS Office и шаблоны документов с включенными в них элементами оформления, стилевой разметкой, средствами автоматизации на основе макросов;
  - разрабатывать макросы, автоматизирующие выполнение типовых операций над документами и сохранять их для дальнейшего использования;
  - дополнять созданные макросы средствами, расширяющими возможности их применения (диалоги с пользователем и организация ветвлений, повторения действий в циклах и пр.);
- *получить навыки*:
  - создания простейших приложений на основе MS Word и MS Excel;
  - разработки интерфейса с пользователем на основе стандартных средств MS Office;
  - реализации средств автоматической генерации и обработки документов MS Office;
  - получения данных из внешних источников, их обработки и включения результатов в документы MS Office.

Содержание занятий

Общий объем аудиторных занятий – 72 часа, что позволяет выдавать слушателям документы государственного образца о повышении квалификации.

Изучение курса предусматривает как аудиторные занятия, так и самостоятельную работу слушателей.

Предлагаемое распределение времени для изучения материала:

- *теоретическая часть курса*: лекции – 12 часов, самостоятельная работа – 24 часа;
- *лабораторный практикум*: аудиторные занятия с преподавателем – 12 часов; самостоятельная работа – 24 часа.

Часы, отведенные для самостоятельной работы, предназначены для изучения теоретического материала, оставленного для самостоятельного изучения, закрепления знаний, полученных на лекциях, и практических навыков по созданию офисных приложений, а также для выполнения итоговых работ по курсу.

Данный курс ориентирован на подготовленных пользователей, поэтому основное время отводится для самостоятельной работы учащихся.

Вопросы для самостоятельного изучения

Для самостоятельного углубленного изучения выделены вопросы, рассмотрение которых в рамках данного курса предусмотрено на уровне основных определений. В учебное пособие по курсу материал по этим вопросам включен в объеме, достаточном для формирования у слушателей общего представления об изучаемом вопросе.

Кроме того, слушателям рекомендуется после проведения лекций самостоятельно проверить полученные знания, отвечая на вопросы, приведенные в конце каждой главы учебного пособия.

Содержание приложений

В приложении приведены описания простейших приложений, которые могут быть созданы опытными пользователями пакета MS Office, и порядка их разработки.

Кроме того, приведено описание пакета, созданного для автоматизации подготовки отчетов на основе приложений MS Office, позволяющее оценить возможности средств разработки и представить основные приемы создания офисных приложений. Пользователи, имеющие программистскую подготовку, могут более углубленно изучить возможности офисного программирования с помощью материала, представленного в книге.

Навыки самостоятельной практической работы по созданию офисных приложений могут быть закреплены при выполнении итоговых работ.

Итоговые работы

При выполнении итоговой работы проверяются и закрепляются знания и навыки, полученные при изучении материала пособия. Кроме того, проверяется способность слушателя работать самостоятельно, изучая новые возможности приложений, используя материал пособия и справочную систему пакета.

Итоговая аттестация может проводиться по результатам собеседования по теоретической части курса и выполнения итоговых работ. Теоретический зачет может быть проведен в форме теста. Практическая часть курса может быть оценена в зависимости от результатов выполнения лабораторных работ или итоговой аттестационной работы.

Введение

Очень многие виды деятельности связаны с трудоемкой, отнимающей массу времени работой по вводу данных, их форматированию, подготовке информации к последующей обработке на персональном компьютере. При этом собственно анализ данных и подготовка отчетов по результатам этого анализа занимает значительно меньше времени, чем предшествующая рутинная работа.

В соответствии с требованиями времени происходит переход от изолированных систем к интегрированным рабочим местам. При этом все чаще используются языки программирования, встроенные в настольные приложения, ориентированные на конечного пользователя.

Для извлечения нужной информации, ее обработки и анализа раньше приходилось ставить соответствующую задачу перед программистами, которые должны были писать программы поиска и извлечения нужной информации. Но не всегда можно откладывать решение задачи на длительное время. Развитие программного и аппаратного обеспечения сделало вполне реальным создание интегрированных рабочих мест. Чтобы интегрированное рабочее место позволяло людям быстрее и лучше выполнять свою работу, оно должно предоставлять своим пользователям возможность быстро извлекать и использовать необходимые данные, любую информацию.

Встроенные средства программирования стали несомненным достоинством большинства настольных приложений. Пакеты офисных приложений превратились в платформы для разработки новых приложений. Фирма Microsoft даже в первые версии своих продуктов не только включила средства создания макросов (записанных последовательностей нажатия клавиш, которые пользователь мог воспроизвести как одну команду) – в Excel язык программирования макросов был дополнен структурами управления, процедурами, функциями и возможностями разработки интерфейсов приложений Windows. Позднее были объединены понятия языка программирования макросов и на-

стоящего языка программирования (Basic).Таким образом появился Visual Basic for Applications (VBA).

В настоящее время пакет Microsoft Office предоставляет пользователям мощные средства разработки приложений. Каждое приложение пакета содержит встроенный язык программирования, который позволяет настроить базовое приложение так, чтобы оно отвечало потребностям его пользователя.

Настольные приложения, которые используются для разработки конкретных пользовательских приложений, называются офисными средствами разработки (Office-based development tool).

В данной работе предлагается описание средств разработки таких приложений на основе семейства Microsoft Office, которые позволили бы пользователям тратить меньше времени на рутинные операции. Использование этих средств не требует глубоких знаний в области программирования: макрос может быть создан средствами записи макросов («программирования без программирования»), включенными в состав приложений Microsoft Office, после чего макрос может быть изменен и дополнен с помощью редактора VBA. Основное внимание уделяется разработке приложений на основе Word и Excel. Представлены также средства доступа к данным из приложений MS Office.

Глава 1. Общая характеристика возможностей VBA

Полномасштабный язык программирования VBA включает следующие средства:

- полный доступ к командам приложения;
- возможности изменения стандартного интерфейса и создания профессионального пользовательского интерфейса;
- наличие традиционного набора структур управления;
- использование функций;
- встроенный текстовый редактор для написания кода;
- средства структурирования кода;
- средства отладки;

– использование Windows API и пользовательских библиотек динамической компоновки (DLL).

Офисные средства не обеспечивают высокой эффективности, но позволяют разработать приложения для бизнеса очень быстро, поскольку используют функциональность, уже заложенную в базовое приложение. Кроме того, внедряются такие приложения очень быстро, так как пользователь остается в привычной среде базового приложения.

Можно выделить следующие основные средства структуризации приложений, созданных на основе MS Office, позволяющие превратить их в естественные расширения приложений пакета:

– разработка процедур, реализующих сложные задачи, часто выполняемые пользователями, с помощью средств макрогенерации приложений пакета или редактора VBA;
– создание собственных диалоговых окон для решения задач поиска и редактирования частей документа, его стандартных элементов, данных;
– добавление пользовательских меню и панелей инструментов, включающих команды, процедуры для реализации которых создаются пользователем (программистом);
– использование процедур обработки событий объектов приложений Office (создания, открытия, закрытия и т.п. файлов);
– замещение стандартных команд приложений Office;
– использование шаблонов и надстроек;
– защита данных от несанкционированного или некорректного доступа, блокирование команд.

Механизмы реализации этих средств могут быть различными для приложений Office. Общие возможности и особенности использования перечисленных возможностей рассматриваются ниже.

Глава 2. Основные принципы разработки приложений

Продуктивность работы возрастает не тогда, когда начинают использовать новый компьютер более совершенной модели, а только после того, как пересматривается отношение к самому процессу работы. Часто сотрудники склонны использовать персональный компьютер как те средства, которые он заменил (например, пишущую машинку, калькулятор и т.п.). Компьютер будет использоваться эффективно только после создания необходимых для работы приложений, ориентированных на конкретных пользователей и конкретное применение, только после переобучения сотрудников, освоения ими новых технологий.

Процесс организации работы должен отвечать некоторым общим принципам:

– *однократный ввод данных и их совместное использование* (информация (названия фирм, фамилии, адреса и т.п.) должна вводиться только один раз и сохраняться в файлах или в базе данных для последующего использования в различных приложениях),
– *превращение компьютера в центральное звено документооборота предприятия*, устранение «устаревших» действий и «лишних» шагов (использование «живых» документов, минимизация «бумажного» документооборота).

Разработка приложений должна начинаться с анализа задач, который можно выполнить как следующую последовательность шагов:

– определение задач;
– определение сценариев для каждой задачи;
– анализ сценариев;
– группировка функциональности по этапам постановки;
– тестирование.

Первый шаг требует определить, кто, что, где, когда и как делает. При этом необходимо идентифицировать не только явно поставленные задачи (в решении которых участвуют пользователи), но и задачи, которые «скрыты» от пользователя. Для этого следует понять процесс бизнеса, для которого разрабатывается приложение.

Требования к приложению описываются в терминах *сценариев*, каждый из которых представляет от начала до конца взаимодействие пользователя с персональным компьютером (ПК) при решении конкретной задачи. В описании сценариев определяются все элементы приложения: формы, команды, сообщения, отчеты. Эти требования должны быть задокументированы и подписаны заказчиком. Данные процедуры являются залогом того, что спецификации не «поплывут»: «требования к приложению подобны воде, на них легче опираться, когда они заморожены».

Цель анализа сценариев состоит в распределении функциональности по компонентам приложения: приложение должно быть разбито на небольшие компоненты, тогда его легче разрабатывать и сопровождать. Код компонентов приложения необходимо комментировать. Если процедура или функция вызывается из других процедур, в комментариях необходимо перечислить их все: это поможет при внесении изменений.

Если приложения разрабатываются на основе базовых приложений, то их поставку можно осуществлять «по частям». Для этого необходимо распределить работу по этапам так, чтобы обновления версий происходили «гладко», т. е. без переучивания пользователя и повторного ввода информации. Поставленная пользователю очередная версия может проходить тестирование во время разработки следующей версии. Полученные от пользователей рекламации дают информацию о том, какие компоненты не работают вовсе, а какие работают не так, как хотелось бы. Ошибки должны быть исправлены, а замечания должны быть учтены при поставке следующей версии приложения.

Разработка приложений является итерационным процессом: они могут дорабатываться и улучшаться постоянно. Лучшие приложения строятся «с прицелом на будущее»: они могут модифицироваться и наращиваться разработчиками, причем нет необходимости все начинать «с нуля».

Вопросы для самопроверки

1. Что означает принцип однократности ввода данных? Какие преимущества он обеспечивает?
2. Каковы этапы создания приложений? Какие задачи решаются на каждом этапе?
3. Что представляет собой сценарий приложения? С какой целью он разрабатывается, какие элементы включает?

Глава 3. Основные правила разработки интерфейса

Для разработки интерфейса пользователя нужно анализировать действия пользователя при выполнении задачи. Интерфейс должен быть «ориентированным на человека» (дружественным). Хороший интерфейс позволяет работать с приложением почти интуитивно.

При создании «правильного» интерфейса следует учитывать следующие принципы:

- быть «последовательными»;
- использовать информативную графику, а не украшательства;
- добиваться легкости чтения;
- показывать, а не говорить;
- делать, а не спрашивать;
- не терять нити диалога;
- разрешать пользователю управлять.

Первый принцип предостерегает от неоправданного «творчества». При разработке интерфейса следует ориентироваться на знания и умения пользователей, которые не должны тратить время на постоянное переобучение, забывать ранее приобретенные навыки. Разработчик может использовать стандартный интерфейс Microsoft Office, «включая» его в свое приложение. В интерфейсе должны действовать стандартные соглашения (все команды доступны через меню, а наиболее часто используемые – через кнопки на панели инструментов).

Информативная графика позволяет людям работать с приложением аналогично тому, как они работали бы с «реальными» документами, поэтому элементы управления, значки должны отражать смысл соответствующих операций (например, лупа должна увеличивать, а ножницы вырезать). Стандартные элементы управления моделируют объекты реального мира, с которыми люди привыкли работать.

Основные элементы управления, организация диалога рассматриваются ниже.

Одна из основных задач при разработке интерфейса – добиться легкости чтения. Эта задача решается выбором наиболее подходящих шрифтов, начертания символов. Кроме того, следует зрительно выделять различные виды текстовой информации: надписи или пояснительный текст следует размещать прямо на форме, но не следует помещать в поле или в рамку; информацию, которая не является постоянной, но которую нельзя редактировать, лучше помещать в текстовое поле, сделав его недоступным и изменив фон; для редактируемой текстовой информации лучше использовать обычное поле ввода и т.п..

Известно, что пользователи не читают инструкций! То есть читают, но только тогда, когда ничего уже не помогает. Следовательно, хороший интерфейс должен сам подсказывать, что делать дальше, он должен вести пользователя, помогать ему решить задачу до конца. Размещение элементов управления должно соответствовать логике приложения. Экран должен быть организован в соответствии со стандартами. Из меню лучше убрать не используемые в приложении команды. Приложение не должно быть «слишком бдительным»: оно не должно задавать лишних вопросов, подтверждение следует запрашивать только тогда, когда какое-либо действие имеет необратимые последствия.

Диалог в приложении должен быть лаконичным. Нужно стараться использовать надписи из одного слова, тексты всплывающих подсказок должны быть краткими (два-три слова), строка состояния должна содержать короткую фразу, поясняющую состояние приложения или выполняемое в данный момент действие. В сообщениях следует использовать стандартные значки.

Хороший пользовательский интерфейс одинаково удобен для всех категорий пользователей: для новичков и опытных пользователей, для тех, кто пользуется мышью и для тех, кто предпочитает клавиатуру. Поэтому:

- все команды должны присутствовать в меню (для новичков);
- для постоянно используемых команд должны быть созданы панели инструментов;

- для часто используемых команд должны быть предусмотрены комбинации клавиш быстрого вызова, которые должны быть указаны в меню;
- следует использовать всплывающие подсказки;
- для подходящих случаев следует определять контекстные меню;
- пользователям должны «прощаться» ошибки: в случае ошибки следует не только выводить сообщение об ошибке, но и подсказку о том, что нужно сделать, чтобы ее избежать.

Для разработки простого в использовании интерфейса нужно выделить задачи, решаемые пользователями, и выписать сценарии их решения. Интерфейс должен позволять достаточно просто решать ежедневные задачи, а также давать возможность работать в исключительных ситуациях.

До начала кодирования следует создать прототип на бумаге, показать пользователю, как будет проходить работа с приложением. Все замечания пользователя можно записать и учесть при кодировании.

Применимость интерфейса желательно проверить на нескольких пользователях, тогда большинство проблем будет обнаружено, и появится возможность их своевременного устранения.

Вопросы для самопроверки

1. Перечислите основные принципы разработки пользовательского интерфейса. Расшифруйте их содержание.
   - Какова цель создания прототипа приложения, макета интерфейса?
   - Какими средствами достигается соответствие основным требованиям, предъявляемым к интерфейсу приложения?
   - Каково назначение подсказок?
   - Какова роль графических изображений в элементах пользовательского интерфейса?

Глава 4. Visual Basic для приложений

VBA является языком программирования пакета Microsoft Office. Имеется языковое ядро Visual Basic, к которому каждый продукт добавляет свою специфическую функциональность. Все используемые средства содержатся в библиотеках (файлы с расширением DLL), которые должны быть установлены и доступны при создании приложений. Каждая библиотека содержит «ресурсы» (константы, программный код и пр.), используемые для разработки приложений. Применение библиотек позволяет разработчикам создавать приложения из готовых «кирпичиков», сохраненных в библиотеке, а не разрабатывать их « с нуля», повторно решая уже решенные ранее задачи, повторяя один и тот же код. Специфические для приложений библиотеки объектов являются важнейшей составной частью VBA (библиотека VBA содержит около 200 функций, свойств и методов и около 200 встроенных констант, а библиотека объектов Excel, обеспечивающая доступ ко всем функциям Excel, – примерно в 15 раз больше). Таким образом, профессиональные разработчики должны быть универсалами, которые знают не только язык программирования и базы данных, но и владеют средствами анализа с помощью электронных таблиц, настольными издательскими системами, методами подготовки документов и презентаций.

Версии Word, Excel и PowerPoint, входящие в состав Microsoft Office, используют редактор Visual Basic (Visual Basic Editor – VBE) и VBA. В Access используется другая версия VBA. Но эти версии достаточно близки.

Разработчик может создавать приложения, использующие возможности нескольких программ пакета. При этом одно из приложений является основным, реализующим для своего пользователя базовую функциональность на основе средств одного из офисных приложений. Другие приложения подключаются для решения вспомогательных задач. Например, основное приложение – MS Excel (с его помощью организуется ввод, хранение в таблицах и обработка (анализ) данных); а для формирования отчетов, документов, в которые выводятся результаты вычислений, вызывается MS Word. При этом основное приложение (Excel) играет роль *клиента*, запрашивая для генерации документов услуги второго приложения (MS Word), которое выступает в качестве *сервера*, обслуживающего запросы.

Общий язык программирования обеспечивает *сквозную программируемость* для всех базовых продуктов. Механизм Automation (*OLE Automation*) позволяет разработчикам относительно просто управлять работой сервера и клиентского приложения и собирать приложения для бизнеса из различных компонентов продуктов Microsoft Office. OLE (Object Linking and Embedding – внедрение и связывание объектов) – технология, позволяющая в документы одного приложения включать объекты, созданные в других приложениях, разрабатывать составные документы; Automation позволяет программным путем обращаться к объектам различных приложений, их свойствам и методам.

Все приложения Microsoft Office включают *средства записи макросов* (команда **Сервис ▸ Макрос ▸ Начать запись…**), позволяющие пользователям «программировать без программирования», но для внесения в записанные макросы изменений, их усовершенствования (например, организации диалога, повторений или ветвлений при исполнении последовательности команд макроса), создания пользовательских функций и т.п. необходимы дополнительные средства. Для написания и модификации кода, разработки пользовательских форм используется отдельное приложение – Редактор Visual Basic.

Вопросы для самопроверки

1. Каково назначение библиотек пакета MS Office?
2. Какими возможностями обеспечивают пользователя средства создания макросов?
3. Как распределяются функции между приложениями-клиентами и приложениями-серверами?
4. Какими возможностями обеспечивает пользователей и разработчиков технология OLE Automation?

Глава 5. Основы алгоритмизации

При решении задач с помощью компьютера пользователю необходимо формализовать описание решаемой задачи, привести его к виду, пригодному для «понимания» вычислительной машиной. Ключевым понятием при этом является понятие алгоритма.

5.1. Определение и свойства алгоритмов

Понятие алгоритма относится к числу основных понятий информатики. В течение длительного времени его употребляли только математики, понимая под алгоритмом *правило решения задач некоторого класса*.

С появлением ЭВМ оно получило очень широкую известность. Постоянное расширение области применения вычислительной техники, огромные перспективы использования компьютеров для решения самых разнообразных проблем привели к становлению новой самостоятельной науки – теории алгоритмов. Разумеется, приведенная формулировка не претендует на роль точного определения понятия *алгоритм*, она лишь поясняет его смысл, выражая то интуитивное воззрение, которое сложилось еще в древности.

Термин *алгоритм* содержит преобразованное географическое название древнего государства в Средней Азии – Хорезм, родины человека по имени Мухаммед ибн Муса аль-Хорезми, ориентировочные годы жизни которого 783-850. Его труды, переведенные в XII веке с арабского на латинский язык, познакомили европейцев с десятичной позиционной системой счисления и с правилами выполнения четырех арифметических действий над многозначными числами. *Формальный характер* этих элементарных операций, которые всеми и всегда выполняются одинаково при полном отвлечении от содержательного смысла операндов, означает, что они могут быть автоматизированы. А имя самого ученого в латинизированной транскрипции *Algorithmi* со временем превратилось в общее название однозначно трактуемой процедуры решения задачи, достижения поставленной цели.

Формирование строгого научного определения алгоритма не закончено и в настоящее время. Современное интуитивное представление таково: *алгоритм* – это конечная последовательность понятных и точных предписаний исполнителю выполнить конечную цепочку действий, приводящих от допустимых исходных данных к искомому результату.

Такая цепочка действий называется *алгоритмическим процессом*, а каждое действие – *шагом* или *операцией*.

Отметим *основные свойства*, присущие любому алгоритму:

1. *Дискретность алгоритма*. Описываемый алгоритмом процесс должен быть разбит на конечное число отдельных указаний, четко отделенных друг от друга конечным ненулевым промежутком времени. Только выполнив требования одной инструкции, можно перейти к выполнению следующей.

2. *Понятность алгоритма*. Алгоритм составляется с ориентацией на определенного исполнителя. У каждого исполнителя имеется свой перечень допустимых предписаний, которые этот исполнитель понимает и может выполнить. Этот перечень называется *системой команд исполнителя* (СКИ). Алгоритм должен включать в себя только те предписания, которые входят в СКИ.

3. *Элементарность шагов алгоритма.* Простейшие (основные, натуральные) операции, выполняемые в соответствии с требованиями алгоритма, зависят лишь от характеристик исполнителя, но не от исходных данных и промежуточных результатов. Например, сравнение в ЭВМ двух чисел, выполнение арифметических операций и т.п., но не сравнение двух файлов, потенциальная длина которых не ограничена.

4. *Точность (однозначность, определенность, детерминированность) алгоритма.* Формулировка алгоритма полностью определяет все действия исполнителя, у которого никогда не должна возникать потребность в принятии самостоятельных решений, не предусмотренных составителем алгоритма. Применяя алгоритм к одним и тем же исходным данным несколько раз, исполнитель получает одну и ту же цепочку промежуточных результатов на каждом шаге, и, соответственно, один и тот же окончательный результат. Результаты не должны зависеть ни от каких случайных факторов.

5. *Массовость алгоритма.* Для каждого алгоритма существует некоторый *класс объектов* (предметов, чисел и т.д.), и *все* они (а не какое-то их количество, конечное, бесконечное или равное нулю) *допустимы* в качестве исходных данных.

6. *Конечность (сходимость, результативность, финитность) алгоритма.* Алгоритмический процесс должен оканчиваться через конечное число шагов, на каждом шаге не должно возникать препятствий для его выполнения, и после остановки можно получить искомый результат. Это требование не учитывает реальных ограничений, связанных с затратами времени и расходованием других ресурсов до завершения алгоритмического процесса за конечное (но заранее неизвестное) число шагов. Поэтому конечность означает лишь *потенциальную* осуществимость алгоритма, хотя на практике, конечно, всегда требуется *реальная* его выполнимость.

Компьютер может «понимать» и исполнять только четко сформулированные команды, заданные в определенной последовательности, поэтому, работая с компьютером, *всегда необходимо следовать инструкциям*, которые содержатся в программной документации, руководствах и справочниках, сопровождающих программы. В этих инструкциях описываются алгоритмы решения задач с помощью компьютера.

Пример алгоритма – инструкция по созданию нового файла в MS Word, записанная на естественном языке:

1. В меню **Файл** выполните команду **Создать…**.
2. Если вы хотите *использовать для создания файла существующие шаблоны документов, мастера или документы*, выполните одно из следующих действий:
   a) Для использования *шаблонов* и *мастеров*:
      – В области задач «Создание документа», в группе «Создание с помощью шаблона» выберите строку «Общие шаблоны».
      – Выберите вкладку, соответствующую типу документа, который требуется создать, и щелкните дважды значок шаблона или мастера, который предполагается использовать.

   b) Для использования *копии существующего документа*:
      – В области задач «Создание документа», в группе «Создание из имеющегося документа» выберите строку «Выбор документа».
      – Выберите в открывшемся окне диалога документ, на основе которого требуется создать новый документ. (Чтобы открыть документ, сохраненный в другой папке, перед выбором документа найдите и откройте нужную папку.)
3. Если Вы хотите создать *новый документ на основе стандартного шаблона*, нажмите кнопку «Создать новый».

Данный алгоритм *понятен* пользователям, имеющим базовые знания и навыки работы в среде Windows. Каждый его шаг *элементарен*: выполняется одна команда с помощью меню или кнопки. Для успешного выполнения команды нужно *точно соблюдать* инструкцию: нельзя пропускать или менять местами шаги. При точном исполнении инструкции за *конечное* число шагов, записанных в ней, всегда создается документ с заданными свойствами и его текст открывается в окне приложения – операция создания *завершается*. Эта инструкция позволяет создавать неограниченное число файлов описанными способами, что характеризует *массовость* алгоритма.

5.2. Основные этапы и методы разработки алгоритмов

Процесс решения любой задачи можно разбить на несколько этапов. Первыми шагами решения всегда являются анализ задачи и разработка (проектирование) алгоритма ее решения.

На этапе анализа задачи уточняется ее постановка, исходные данные для ее решения и предъявляемые к решению требования и условия, при которых задача должна быть решена.

Разрабатывая алгоритм решения задачи, необходимо ответить на вопросы:

– Существуют ли решенные аналогичные задачи?

– Какие структуры данных лучше всего подходят для решения задачи?

Первое, что требуется от алгоритма, это правильно реализовать функцию, которая каждому элементу из множества исходных данных ставит в соответствие возможный результат. И второе: от алгоритма требуется такая реализация этой функции, при которой время решения и затрачиваемые на него усилия были бы по возможности минимальными.

На сегодняшний день, по-видимому, самой популярной методикой проектирования алгоритмов, уменьшающей вероятность ошибок, упрощающей понимание и облегчающей их модификацию, считается *технология нисходящего структурного проектирования*, или *проектирования сверху вниз*.

Этот метод связан с *последовательной детализацией* решения, со сведением поставленной задачи к последовательности более простых задач, которые легче поддаются решению, чем исходная, но из их решений может быть получено решение первоначальной задачи. При использовании этого метода осуществляется *декомпозиция* общей задачи на точно определенные подзадачи. Затем для полученных подзадач также повторяются процессы декомпозиции, которые повторяются до получения подзадач настолько простых, что их решение может быть сформулировано в терминах элементарных операций, понятных исполнителю алгоритма.

По мере того как алгоритмы становятся все более сложными, растет трудность понимания того, как они работают, доказательства их правильности, исправления обнаруженных в них ошибок, внесения изменений. Поэтому очень важно правильно выполнить проектирование алгоритма и выбрать подходящий метод (методы) решения задачи (или подзадач, на которые при проектировании распалась исходная задача).

Для записи алгоритма в процессе решения задачи используются различные способы.

5.3. Основные способы описания алгоритмов

Выбор способа описания алгоритма зависит от предполагаемого способа его реализации, от исполнителя. Кроме того, существуют различные методологические подходы к разработке алгоритмов, которые также определяют, каким образом алгоритм будет описан.

Словесное описание – это описание на естественном языке, когда исполнителем алгоритма является человек. Процесс описания алгоритма в этом случае предполагает получение ответов на ряд полезных вопросов, например следующих:

– Понятна ли терминология исполнителю?
– Что дано в качестве исходных данных?
– Что нужно найти (что будет результатом выполнения)?
– Как определить решение?
– Каких данных не хватает?
– Являются ли какие-то имеющиеся данные бесполезными?
– Какие сделаны допущения?

Возможны и другие вопросы в зависимости от конкретной задачи. Некоторые вопросы приходится ставить повторно.

Примерами словесных описаний алгоритмов решения задач с помощью компьютера являются инструкции по использованию команд, приведенные в справочных системах различных приложений.

По мере совершенствования информационных технологий разрабатывались и развивались специальные методы, позволяющие записать алгоритмы, четко формализовать их описания.

Наиболее распространенным способом описания алгоритмов является описание алгоритмов в виде структурных *блок-схем*.

Графическое представление алгоритмов решения задач в различных областях с помощью специальных графических нотаций, *диаграмм*, принято как при разработке информационных систем, при анализе и моделировании деятельности предприятий и организаций, так и в других случаях, когда необходимо формально описать деятельность предприятий (например, при сер-

тификации предприятия). Примеры таких описаний – диаграммы потоков работ (*Workflow Diagram*), функционального моделирования (*IDEF0 Diagram*) и т.п.

Как только алгоритм описан и есть уверенность в его правильности, наступает этап реализации алгоритма.

Если для решения задачи разрабатывается специальная *программа*, то на этой стадии осуществляется написание кода программы для компьютера на одном из искусственных языков.

5.4. Основные управляющие структуры, применяемые для записи алгоритмов

При записи алгоритмов используются различные управляющие структуры, обеспечивающие возможность записи последовательности действий, включенных в алгоритм.

Простейший вариант алгоритма предполагает последовательное выполнение всех его шагов. Управляющая структура, представляющая такой порядок действий, – *цепочка*. Все шаги алгоритма, предписанные ими действия выполняются строго последовательно без каких-либо вариантов выбора действий, условий, определяющих их выполнение. Например, сохранение с помощью команд меню созданного файла в заранее подготовленной на внешнем запоминающем устройстве папке предполагает в приложениях MS Office выполнение следующих шагов:

1. Открыть меню **Файл**.
2. Выбрать команду **Сохранить как…**.
3. В диалоговом окне команды с помощью элемента «Папка» выбрать и открыть заданную папку для размещения созданного файла.
4. В диалоговом окне команды в поле ввода «Имя файла» ввести имя сохраняемого файла.
5. Завершить выполнение операции щелчком по кнопке **Сохранить** в диалоговом окне команды.

В блок-схеме для представления отдельных шагов (действий) алгоритма используется специальный *блок* – прямоугольник, в который вписывается соответствующая операция (рис. 5.1).

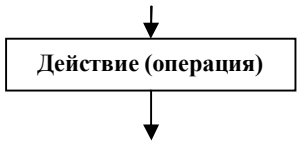


Рис. 5.1. **Представление одного шага алгоритма**

Дуги (направленные соединительные линии со стрелками) показывают последовательность выполнения операций, объединенных в цепочку.

Описанные выше действия по сохранению файла могут быть представлены, как это показано на рис. 5.2.

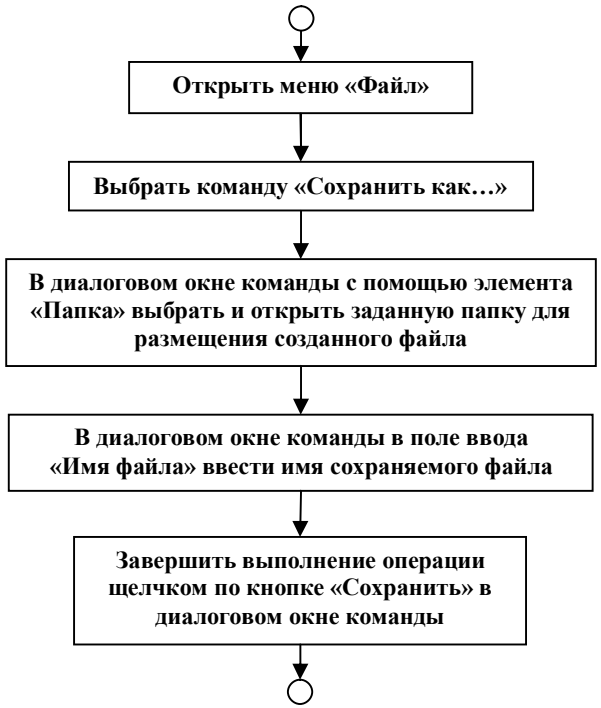


Рис. 5.2. **Последовательность (цепочка) операций, которую нужно выполнить для сохранения файла**

Однако при разработке более сложных алгоритмов возникает необходимость учитывать условия выполнения отдельных шагов, зависимости, которые возникают при реализации отдельных действий. Для представления таких алгоритмов, в которых некоторые из шагов обусловлены каким-либо образом, в которых существуют различные варианты действий, а выбор конкретного варианта определяется каким-либо условием, применяются другие управляющие структуры, представляющие *ветвления*. Конструкция ветвления позволяет при выполнении алгоритма сделать *выбор* между несколькими решениями в зависимости от значения некоторого параметра или параметров, являющихся входными данными алгоритма или вычисляемыми в ходе его выполнения (на предшествующих ветвлению шагах).

Существуют различные варианты реализации ветвлений: альтернатива и многозначные ветвления.

Альтернатива – простейшая форма ветвления, где есть два возможных пути выполнения алгоритма и выбор зависит от того, верно или неверно некоторое условие (рис. 5.3).

***Рис. 5.3*. Представление ветвления (альтернативы) в блок-схеме**

В соответствии с приведенной схемой, *если* верно *условие*, определяющее выбор решения, *то* выполняется *Вариант 1* действий, определенных алгоритмом, *иначе – Вариант 2*.

Для обеспечения большей «читабельности» программы, простоты понимания алгоритма в каждую управляющую структуру должен быть только один вход и один выход из нее. После выбора и выполнения одного из вариантов действий выполнение алгоритма продолжается со следующего за ветвлением шага.

В некоторых случаях при невыполнении условия никакие действия выполняться не должны. Тогда используется «сокращенный вариант ветвления» (рис. 5.4).

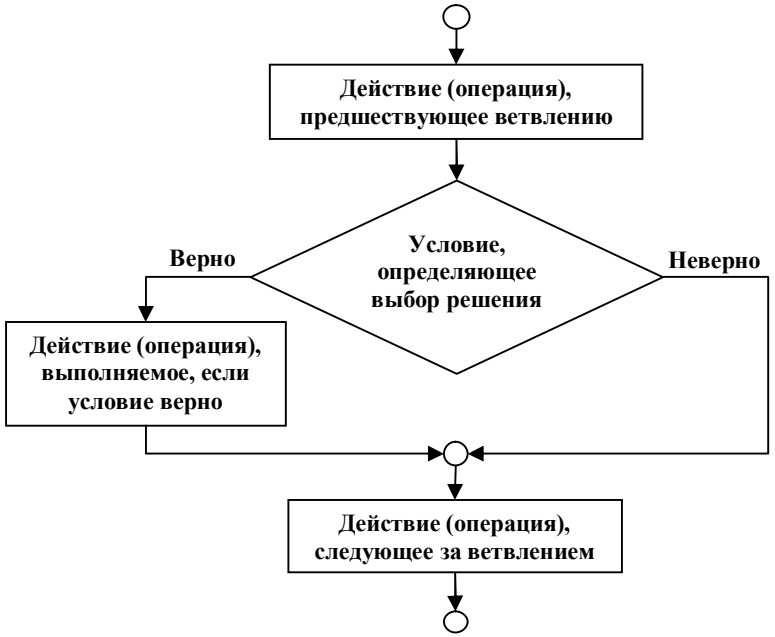


***Рис.5.4*. Представление ветвления без альтернативы в блок-схеме**

Во многих случаях при решении задачи выбирать приходится не из двух, а из большего числа альтернатив. Такие конструкции выбора поддерживаются практически во всех языках программирования. Конструкции могут быть *вложенными*.

Управляющие структуры языка VBA рассматриваются ниже.

Еще одна важная управляющая структура – *повторение*, или *итерация*, *цикл*. Эта конструкция используется для сокращения записи алгоритма, если некоторые действия или последовательность действий в нем *повторяется многократно*.

В зависимости от того, каким образом определяется число повторений, различают несколько *типов циклов*: циклы*, управляемые условием* (*с предусловием* (рис. 5.5) и *с постусловием* (рис. 5.6)); циклы, *управляемые параметром* (они обычно могут рассматриваться как частный случай циклов с предусловием). Операции, повторяющиеся в цикле, называют *телом цикла*. Тело цикла может включать несколько действий, повторяться некоторое число раз. Число повторений определяется по-разному в разных видах циклов.

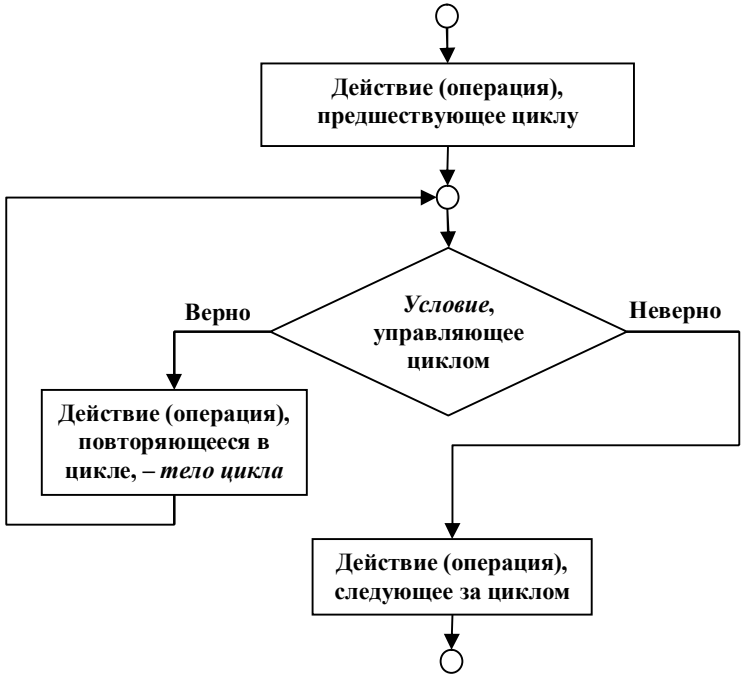


Рис. 5.5. **Представление цикла с предусловием в блок-схеме**

В соответствии со схемой, показанной на рис. 5.5, действия, включенные в тело цикла с предусловием, повторяются, *пока* сохраняет силу (выполняется, является верным) условие, управляющее циклом.

После выполнения операций, предшествующих циклу, *перед входом* в цикл (отсюда и название – цикл с предусловием) проверяется, истинно ли (верно ли) условие. Если условие неверно, то цикл не будет выполняться ни разу. Если условие верно, то выполняется действие, включенное в тело цикла (повторяющееся в цикле). При выполнении этого действия (или целой последовательности действий – тела цикла) могут измениться некоторые параметры, влияющие на истинность условия, управляющего циклом, поэтому по завершении выполнения тела цикла снова осуществляется проверка истинности этого условия.

Эта последовательность проверок условия и действий, включенных в тело цикла, повторяется, пока не станет ложным (неверным) заданное условие, т.е. конструкция имеет смысл: «*Пока верно условие повторять тело цикла*».

Другой разновидностью цикла с предусловием является цикл, в котором тело повторяется до тех пор, пока условие, управляющее циклом, неверно. Как только значения параметров в цикле изменятся так, что условие станет верным, повторение действий, включенных в тело цикла, прекращается.

При использовании в алгоритме цикла с постусловием действия, входящие в тело цикла, обязательно (безусловно) выполняются хотя бы один раз и только потом осуществляется проверка условия, управляющего циклом. На рис. 5.6 показан вид схемы алгоритма с постусловием, в котором тело цикла повторяется до тех пор, *пока не* станет верным. Как только условие становится истинным, повторения прекращаются, происходит переход к выполнению действий, следующих за циклом. Другая разновидность цикла с постусловием предполагает, что тело цикла повторяется, пока управляющее циклом условие истинно. От одной разновидности цикла всегда можно перейти к другой, заменив условие на противоположное.

Основная проблема при использовании циклов такого типа – определение условий, управляющих повторениями и операций, влияющих на условия. При неправильно заданных условиях

есть опасность, что цикл «не остановится», т.е. произойдет зацикливание программы, алгоритм «потеряет» свойство конечности.

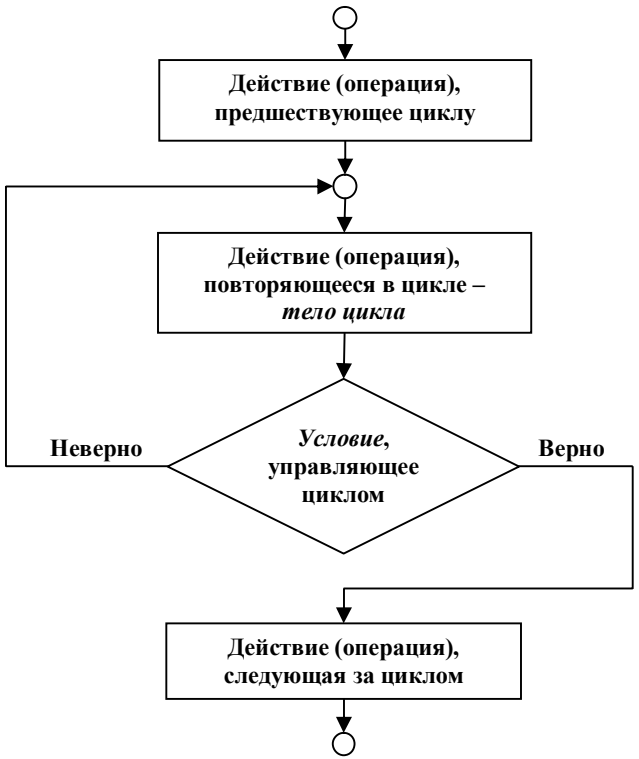


Рис. 5.6. **Представление цикла с постусловием в блок-схеме**

В VBA поддерживается возможность использования всех разновидностей циклов с предусловием и постусловием.

Представление алгоритмов в графическом виде, в форме блок-схем и диаграмм является наиболее наглядным, обеспечивающим легкость восприятия. Но для выполнения алгоритма решения задачи с помощью компьютера необходимо записать его на языке программирования, представить в виде программы.

5.5. Создание программы

Для того чтобы сделать разработанный алгоритм «понятным» для выполнения на компьютере, его нужно записать с помощью *языка программирования*. Таким языком при создании приложений на основе пакета MS Office является VBA.

Описание алгоритма решения задачи на языке программирования представляет собой *программу*. Основное назначение программы – описание используемых при решении задачи данных (исходных и промежуточных данных, результатов вычислений) и шагов алгоритма в виде, пригодном для ввода в компьютер. Описываемые в программе данные и операции полезно комментировать, чтобы при необходимости проще было вносить изменения в текст программы. Хорошо откомментированная программа является «самодокументированной», включает в себя документацию, которая необходима для ее сопровождения, доработки и настройки.

При создании программы необходимо очень тщательно следить, чтобы программа на языке программирования, являющаяся реализацией правильного алгоритма, описанного в словесной форме на первых этапах решения задачи, также была правильной, проверенной и оттестированной. Поэтому эксплуатации программы предшествует ее *отладка*, выявление и исправление синтаксических, семантических и логических ошибок, *тестирование* в различных условиях на широком диапазоне значений исходных данных.

Весь процесс разработки и реализации алгоритма решения задачи должен сопровождаться *документированием* важной информации, связанной с различными этапами разработки, с принимаемыми на этих этапах решениями. Отсутствие документации может привести к проблемам при разрешении конфликтных ситуаций, которые могут возникнуть между разработчиком программы (исполнителем) и ее пользователем (заказчиком). Как и в любой другой области, в сфере информационных технологий, связанных с разработкой программных продуктов, невозможно требовать реализации решений, которые не были должным образом оформлены (документированы).

Вопросы для самопроверки

1. Что такое алгоритм? Приведите различные варианты определения этого термина.
2. Что такое шаг алгоритма?
3. Поясните основные свойства алгоритмов.
4. В чем суть нисходящего структурного проектирования алгоритма? Как вы определите пошаговую детализацию?
5. Сделайте словесное описание известного Вам алгоритма.
6. Какие основвые управляющие структуры используются при записи алгоритмов?
7. Какие конструкции ветвления вы знаете? Опишите их.
8. Представьте в виде блок-схемы алгоритм создания файла, словесное описание которого приведено выше.
9. Приведите примеры известных вам алгоритмов, в которых есть ветвления. Представьте алгоритмы в виде блок-схем.
10. Какие конструкции повторений вы знаете? Опишите их.
11. Чем отличаются циклы с постуслоиями от циклов с предусловиями? В каком случае тело цикла может не выполняться ни разу?
12. Покажите, как от одной разновидности цикла перейти к другой при решении одной и той же задачи? Приведите примеры, когда нельзя использовать цикл с постусловием.
13. Приведите примеры известных вам алгоритмов, в которых есть повторения. Представьте алгоритмы в виде блок-схем.
14. Что такое программа?
15. Какие средства используются для записи программ?
16. Каковы основные шаги разработки программы? Определите цели отладки и тестирования программ.
17. Каково назначение документации и документирования программы?

Глава 6. Введение в язык и редактор Visual Basic

На начальном этапе разработки приложения выбирается основное приложение, являющееся базовым для создания нового проекта. В качестве основной выбирается программа, в которой можно выполнить большую часть требуемых операций. Данная программа будет использоваться для запуска проекта. Для создания проекта используется редактор VBA.

Для вызова редактора есть несколько способов:

- Выполнение команды **Макрос ▸ Редактор Visual Basic** в меню **Сервис**. При этом осуществляется «первый» вызов редактора.
- Запуск редактора щелчком по кнопке **Редактор Visual Basic** на панели инструментов «Visual Basic».
- Для редактирования текста ранее записанного макроса нужно выполнить команду **Макрос ▸ Макросы…** в меню **Сервис**, выбрать макрос из списка и щелкнуть кнопку **Изменить**.

6.1. Окно Редактора Visual Basic

При запуске редактора в основном его окне появляется три окна: окно проекта, окно свойств и окно программы (рис. 6.1).

Окно проекта, расположенное вверху слева, позволяет управлять различными элементами проекта VBA, в том числе блоками кода и диалоговыми окнами. В этом окне отображаются несколько папок. Первая содержит объекты, специфичные для приложения Office, из которого запущен редактор (например, Microsoft Excel или Microsoft Word). Вторая содержит пользовательские диалоговые окна и формы (она называется «Формы» или «UserForms»). Третья папка содержит пользовательские процедуры, сгруппированные в модули (она так и называется «Модули» или «Modules»). Последние две папки появляются, если созданы модули (например, был записан макрос) или формы. Формы и модули могут быть созданы с помощью соответствующих команд меню **Вставка**. Папка «Ссылки» содержит имеющиеся в документе ссылки (например, на шаблон).

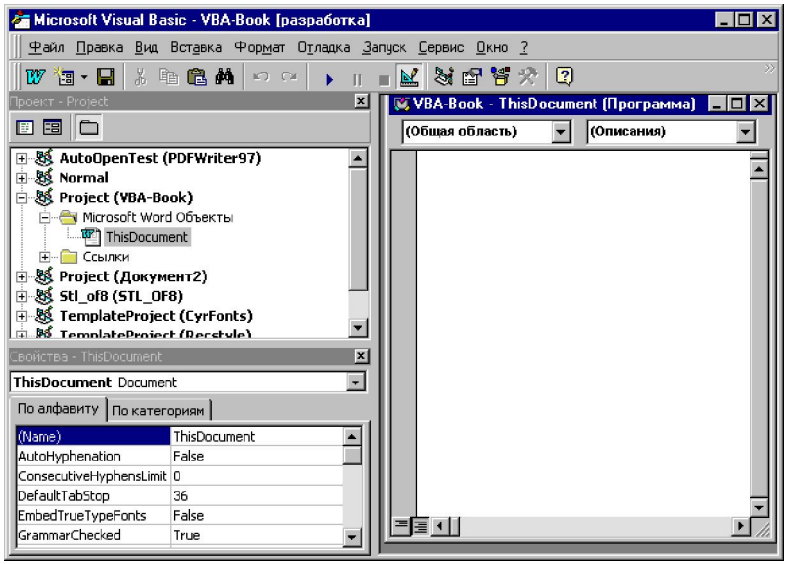


Рис. 6.1. **Окно редактора Visual Basic**

Если это окно отсутствует, его можно открыть с помощью команды меню **Вид**.

В *окне свойств*, расположенном слева внизу, отображаются все характеристики объектов – элементов проекта VBA. Если это окно отсутствует, его также можно открыть с помощью команды меню **Вид**. Для просмотра и изменения свойств объекта его нужно выделить в окне проекта. После этого в окне свойств будут показаны его свойства, их можно просмотреть и изменить, выбрав из списков или введя с клавиатуры нужные значения.

Справа расположено *окно программы* (окно редактора кода, окно модуля). Это окно используется при написании любой программы на VBA. При реализации проекта создаваемые подпрограммы можно сохранить отдельно, сгруппировав их по назначению и связям в нескольких *модулях*. Для создания нового модуля нужно выделить в окне проекта проект, в который требуется добавить модуль; в меню **Вставка** выбрать строку **Модуль** (появится пустое окно модуля); в окне свойств задать имя созданного модуля. Окно существующего модуля можно открыть через меню **Вид**. Команда **Программа** доступна в нем, если в окне проекта выделен модуль.

Модуль состоит из *описаний* (инструкций, используемых для описания переменных, констант, типов данных, ссылок на внешние функции, хранящиеся в библиотеках динамической компоновки) и *кода процедур* приложения (процедура представляет собой поименованный набор инструкций). *Инструкции* – это единицы кода, представляющие операции, описания или области определения.

Окно программы имеет два режима просмотра: представление полного модуля и представление процедуры. Переключение режимов осуществляется с помощью кнопок в левом нижнем углу окна программы. В режиме представления процедуры в каждый момент времени в окне отображается только одна процедура. Для ее выбора можно воспользоваться раскрывающимися списками, которые находятся в верхней части окна программы: в левом списке перечисляются все объекты, содержащие процедуры, а в правом – процедуры выделенного объекта. В режиме просмотра полного модуля в окне можно пролистать все процедуры и функции, описанные в модуле.

Вместо окна программы может быть открыто *окно просмотра объектов* или *окно редактора пользовательских форм* (например, при работе с пользовательской формой можно в режиме «Объект» увидеть форму с размещенными на ней объектами (элементами управления) или в режиме «Программа» просмотреть и отредактировать код, соответствующий выделенному объекту). Переключение режимов выполняется с помощью кнопок, расположенных в верхней части окна проекта, или с помощью команд меню **Вид**. Редактор VBA дает возможность открыть несколько окон для редактирования кода нескольких модулей. Между окнами можно переключаться с помощью команд меню **Окно**.

По мере ввода текста в окне программы Редактор Visual Basic отображает подсказки (синтаксис свойств, методов, операторов и функций по мере их ввода). В некоторых случаях предлагается выбор из списка значений, которые можно ввести. Включение и выключение такого режима выполняется при настройке среды пользователя через команду **Сервис ▸ Пара-**

метры. На вкладке **Редактор** можно установить флажки, управляющие видом окна и параметрами программирования.

При работе с VBA могут быть открыты также окно отладки, окно локальных переменных и окно контрольных значений, используемые при отладке. Переключаться между окнами можно с помощью мыши или через команды меню **Окно**.

Окно программы редактора может выглядеть иначе, так как описанные выше окна могут быть перемещены на другие места или закрыты, их размеры могут быть изменены. Каждый разработчик может настроить их «под себя».

При редактировании текста программы можно получать подсказки о назначении и использовании конкретных процедур, встроенных в VBA, констант, переменных. Для этого нужно выделить нужное имя в тексте программы и нажать клавишу *F1*. На экране откроется соответствующий раздел справочной системы.

Чтобы закрыть Редактор Visual Basic, можно воспользоваться командой **Закрыть и вернуться в …** меню **Файл**.

6.2. Основные правила записи программ на VBA

При создании приложений с помощью офисных средств разработки используются следующие основные понятия:

- процедуры (подпрограммы и функции),
- строки кода,
- комментарии,
- объекты, свойства и методы,
- позиционные и именованные аргументы,
- встроенные константы,
- типы данных,
- объектные переменные,
- структуры управления VBA.

Любая программа на VBA представляет собой последовательность инструкций. Инструкция Visual Basic является полной командой. Она может содержать ключевые слова, символы операций, переменные, константы и выражения.

Любая инструкция относится к одной из следующих категорий:

- Инструкции *описания*, именующие переменные, константы, процедуры, типы данных, а также задающие их характеристики.
- Инструкции *присвоения*, которые присваивают значение выражения переменной или константе.
- *Исполняемые* инструкции, которые инициируют действие. Они могут выполнить метод или функцию, а также могут организовать повторение или ветвление блоков программы. Эти инструкции часто содержат математические или условные операторы.

При описании синтаксических правил VBA полужирным шрифтом показаны ключевые слова VBA, а курсивом – синтаксические обозначения, которые при вводе текстов программ должны заменяться на конкретные значения, имена и т.п. В фигурных скобках записываются возможные альтернативные варианты, при этом разделителем между перечисленными альтернативами служит вертикальная черта (например: { ***While*** | ***Until*** }), из всех указанных вариантов при записи инструкции выбирается один вариант.

При работе с VBE можно использовать режим контроля синтаксических ошибок. В Visual Basic имеется функция проверки синтаксиса: каждая вводимая инструкция проверяется на ошибки синтаксиса, такие как неправильное написание ключевого слова или отсутствующий разделитель (выдается оповещение об обнаруженных ошибках). Если же синтаксис правильный, программа переводится во внутреннюю форму, что ускоряет переход ко времени выполнения. Эта функция включается в начале, но затем ее можно выключить, если предпочтительнее вводить программу без оповещения о возникающих ошибках. Чтобы включить или выключить проверку синтаксиса нужно выбрать команду **Параметры** в меню **Сервис**; перейти на вкладку «Модуль»; установить или сбросить флажок «Проверка синтаксиса»; нажать кнопку **OK**.

6.2.1. Описания в программах на языке VBA

Инструкции описания используются для описания процедур, переменных, массивов и констант и для присвоения им имен. Все переменные, константы, процедуры и функции имеют область действия, где они доступны. При описании процедур, переменных или констант задается также их область определения, которая зависит от того, где расположено описание и какие ключевые слова при этом использованы.

В VBA существует три уровня областей действия переменных:

– уровень процедуры,
– уровень модуля и
– уровень проекта.

Эти уровни применимы и для массивов. Описания типов данных, определенных пользователем, которые должны располагаться в разделе описаний модуля VBA, по умолчанию имеют область действия уровня проекта.

Константы имеют те же три уровня действия, а процедуры – только два: модуля и проекта.

Описания представляют собой неисполняемые программные инструкции, в которых определяются имена процедур, констант или переменных и задаются их характеристики (такие как тип данных).

При разработке приложений отдельные задачи можно выделить и оформить в виде процедур, сгруппированных в модули. *Модуль* включает в себя процедуры, которые описывают выполнение приложением конкретных задач. Процедуры используются для разбиения сложных программистских задач на более простые элементы.

Описания включаются в модули форм, в стандартные модули и в модули класса. Чтобы ввести описания уровня модуля, надо перейти в раздел описаний модуля. Чтобы ввести глобальные описания, надо перейти в раздел описаний модуля и использовать инструкцию ***Public*** для констант и переменных. Для создания описаний можно также воспользоваться ключевыми словами ***Dim***, ***Static*** и ***Private***.

Можно также вводить описания на уровне процедуры. Для

любого программного уровня и для любого способа описания переменных или констант следует применять специальные правила, связанные с областями определения.

Чтобы открыть раздел *описаний модуля*, нужно выбрать в окне проекта модуль формы, стандартный модуль или модуль класса, который надо открыть, и выполнить команду **Программа** в меню **Вид** или контекстном меню, а затем выбрать элемент «(Общая область)» в списке «Объект», расположенном над окном программы. В находящемся справа списке «Процедуры» автоматически будет выведена область окна «(Описания)». Далее можно ввести описание или несколько описаний.

В описании указывается имя, которое будет использоваться для идентификации описываемой процедуры, параметра, переменной, константы или типа. При присвоении имен в модуле Visual Basic используются следующие правила:

– Имена должны начинаться с буквы.
– Имя не может содержать пробел, точку (**.**), восклицательный знак (!) или символы @, &, $, #.
– Имена не должны содержать более 255 символов.
– Как правило, не следует использовать имена, совпадающие с названиями функций, инструкций, и методов языка Visual Basic, так как при этом прекращается выделение в тексте одноименных ключевых слов языка. Чтобы использовать встроенные функции языка, инструкции или методы, имена которых конфликтуют с присвоенным пользователем именем, их необходимо явно указывать. Для этого перед именем встроенной функции, инструкции или метода, должно стоять имя связанной с ними библиотеки типов. Например, если имеется переменная с именем Left, то функция Left должна вызываться как VBA**.**Left.
– Не допускается использование повторяющихся имен на одном уровне области определения.

В языке Visual Basic не различаются строчные и прописные буквы, в инструкции описания сохраняются прописные буквы.

6.2.2. Структуризация программ VBA и правила записи кода процедур

Основной структурной единицей при написании программ на VBA является процедура.

Процедура представляет собой описание *последовательности инструкций* (команд, или операторов), задающих алгоритм решения задачи. Эта последовательность команд записывается всего один раз, но может выполняться там, где это необходимо. Описанные в процедуре операторы образуют *код процедуры*.

Любая инструкция VBA помещается в отдельной строке кода. Если запись команды оказывается слишком «длинной» и «не вписывается» в одну строку в окне программы, что затрудняет просмотр кода, ее можно разбить на несколько строк с помощью символов продолжения строки кода ‘ _’ (в конце строки, которая должна быть продолжена, ставятся пробел и символ подчеркивания). Если строка заканчивается на эти символы, следующая строка рассматривается как ее продолжение.

Для большей наглядности перед началом каждой инструкции ставятся пробелы (задается отступ с помощью табуляции). Отступы позволяют зрительно структурировать программу на экране, подчеркнуть вложенность команд в процедурах.

Инструкции и описания в процедурах сопровождаются комментариями. Комментарием считается текст, следующий за одиночным апострофом (символ ') до конца строки. Даже при записи макроса автоматически генерируется комментарий по информации, введенной в диалоговом окне «Запись макроса».

Самой «простой» инструкцией VBA является команда, имеющая вид:

ИмяПеременной = Выражение

или

Let **ИмяПеременной = Выражение**

Она осуществляет пересылку значения выражения, записанного в правой его части (после знака ‘=’), в переменную, имя которой указано в левой части оператора (перед знаком ‘=’). Тип значения выражения должен соответствовать типу переменной (если он задан явным описанием). В некоторых случаях автоматически выполняется преобразование типа. Полный синтаксис оператора предусматривает использование инструкции ***Let***, но инструкция ***Let*** необязательна и чаще всего опускается.

Для присвоения объекта переменной, описанной как объект, применяется инструкция ***Set***. Ключевое слово ***Set*** обязательно. Инструкции, задающие значение свойства объекта, также являются инструкциями присвоения. Более детально объекты и их свойства рассматриваются ниже, так как набор объектов является специфичным для каждого приложения.

Более подробную информацию можно получить в справочной системе VBA (раздел «Инструкции присвоения»).

Другие инструкции, стандартные структуры управления (ветвления, циклы) описаны ниже.

Команды, включенные в процедуру, выполняются, когда к ней происходит *обращение* (вызов процедуры). Для идентификации процедуры при ее вызове ей присваивается *имя*, которое указывается в *заголовке процедуры*. Все имена в VBA состоят из одного слова – идентификатора и представляют собой последовательность букв и цифр (в имени не должно быть разделителей).

Процедура может вызываться в качестве макроса или функции из приложения Office.

6.2.3. Использование параметров

Для повышения гибкости использования процедур, обеспечения возможности их настройки на различные условия выполнения в описании процедуры могут быть заданы *формальные параметры*, которые являются фактически лишь обозначениями исходных данных, передаваемых в процедуру для обработки, и результатов ее выполнения. Список параметров задается в заголовке процедуры вслед за ее именем.

При вызове процедуры указывается ее имя и, если необходимо, список *фактических параметров* (*аргументов*), которые будут подставлены вместо описанных в процедуре формальных параметров и будут использованы при выполнении всех инструкций, включенных в код процедуры. Передаваемые в процедуру аргументы определяют реальные данные, задача обработки которых решается при обращении к процедуре.

Между формальными и фактическими параметрами при вызове процедуры должно быть установлено соответствие. По способу задания этого соответствия различают позиционные и именованные аргументы (параметры).

Позиционные аргументы при вызове процедуры задаются в строго определенном порядке (этот порядок соответствует порядку соответствующих формальных параметров, заданных в списке при описании процедуры), их значения разделяются запятыми. Если значение какого-либо аргумента при обращении к процедуре не задается (это допускается, когда аргумент является необязательным), вместо него ставится запятая, если только пропущенный параметр не является последним в списке. Например:

```
Answer = InputBox ("Введите имя: ", "Кто Вы? ", , 100, 100)
```

показывает обращение к стандартной процедуре InputBox, где используется передача позиционных аргументов. В середине списка при вызове процедуры пропущен необязательный аргумент, задающий значение, выводимое по умолчанию в качестве результата, а в конце списка пропущены необязательные параметры, задающие подсказки при работе с диалоговым окном. При этом сложно понять назначение каждого аргумента (можно обратиться за помощью к справочной системе).

Именованные аргументы позволяют указывать при вызове процедуры значения в произвольном порядке: соответствие между формальными и фактическими параметрами задается по имени. Ниже приведен пример обращения к процедуре Answer, где соответствие между формальными параметрами и передаваемыми в процедуру аргументами устанавливается по имени (символ подчеркивания в конце строки означает, что в следующей строке содержится продолжение оператора):

```
Answer = InputBox   (Prompt := "Введите имя: ",  _
                     Title :=  "Кто Вы? ",  _
                     Xpos := 100,  _
                     Ypos := 100)
```

Такой способ передачи аргумента является более информативным при чтении текста программы.

Для *передачи аргументов* в процедуру существует два способа: передача параметров *по ссылке* и *по значению*.

При передаче аргумента *по значению* создается копия исходного значения (значения фактического параметра) и это значение присваивается соответствующему формальному параметру. Изменения аргумента в процедуре не распространяются на значение переданного фактического параметра. Для того чтобы указать, что аргумент должен передаваться по значению, перед его именем в списке формальных параметров вставляется ключевое слово ***ByVal***.

При передаче аргумента *по ссылке* копия исходного значения не создается. Соответствующий формальный параметр становится лишь обозначением фактического параметра при выполнении кода процедуры. Все выполняемые над ним действия выполняются над указанным фактическим параметром. Таким образом, в результате выполнения процедуры исходное значение аргумента может быть изменено. Данный способ передачи аргумента задается при описании процедуры с помощью ключевого слова ***ByRef***, записываемого перед именем формального параметра в заголовке процедуры.

По умолчанию аргументы передаются в процедуры по ссылке. Это эффективно, так как все аргументы, передаваемые по ссылке, требуют одинакового времени для передачи и одинакового количества памяти (4 байта) для передачи ссылки (адреса), независимо от типа данных аргумента. При этом следует учитывать, что аргумент, передаваемый в процедуру по ссылке, должен быть того же типа, что и соответствующий формальный параметр.

Если включить ключевое слово ***ByVal*** в описание процедуры, то аргумент будет передан по значению. Аргументы, переданные по значению, занимают от 2 до 16 байт на процедуру, в зависимости от типа данных аргументов. Типы данных большего размера требуют на передачу немного больше времени. По этой причине обычно не следует передавать по значению, например, строки.

При описании параметров в процедуре может быть использовано ключевое слово ***Optional***, которое применяется для обозначения тех аргументов, которые не являются обязательными. Если какой-либо параметр не является обязательным, то и все аргументы, следующие за ним в списке формальных параметров,

также будут необязательными. Все необязательные параметры имеют тип Variant. Ключевое слово ***Optional*** записывается перед именем параметра и описанием способа его передачи.

Ключевое слово ***ParamArray*** используется для принятия в процедуре произвольного числа аргументов и упаковки их в массив переменных типа Variant. Такое описание можно использовать только для последнего параметра в списке. Более детальную информацию можно получить в справочном руководстве VBA.

В программах на языке VBA используется несколько видов процедур: подпрограммы, функции, свойства. Процедуры реализуют обработку событий. Более подробно средства создания и вызова процедур рассматриваются ниже.

6.2.4. Типы данных VBA

VBA поддерживает многочисленные типы данных. Стандартными типами являются ***Boolean*** (логические значения), ***Byte***, ***Integer***, ***Long*** (целочисленные значения), ***Double***, ***Single*** (действительные числа с плавающей точкой), ***Currency*** (денежные значения), ***Date*** (значения даты и времени), ***String*** (строковый тип), ***Object*** (объектные переменные), ***Variant*** (тип, к которому относятся все переменные, не описанные явно с другим типом данных). Пользователь может определить свой тип данных с помощью ключевого слова ***Type***.

Подробную информацию об использовании различных типов, их совместимости можно получить в справочном руководстве по Visual Basic (раздел «Типы данных»).

6.2.5. Описания переменных

Переменные в VBA используются для управления данными, для обозначения значений, которые могут изменяться при выполнении программы. Они нужны, чтобы запоминать результаты вычислений, которые могут потребоваться при выполнении последующих инструкций, если эти значения используются многократно. Прежде чем использовать переменную следует ее описать, чтобы VBA мог выделить место для размещения переменной в памяти и контролировать правильность ее использования.

Описывать переменные в VBA не обязательно, но лучше это делать, так как явные описания позволяют избежать многих ошибок, повысить эффективность программы. Поэтому, чтобы поддерживать хороший стиль программирования, рекомендуется установить при работе с VBA режим, требующий обязательного описания переменных. При работе в таком режиме в начало каждого модуля помещается оператор

Option Explicit

Этот режим задается при установке параметров с помощью соответствующей команды меню **Сервис**. В открывшемся диалоговом окне нужно установить флажок «явное описание переменных».

Неописанные переменные приобретают тип данных Variant. Этот тип данных упрощает написание программ, но его использование может снизить эффективность программы.

Следует предусмотреть применение других типов данных, если

- программа имеет большой размер и очень много переменных;
- требуется как можно более быстрое выполнение программы;
- выполняется прямая запись данных в файлы с произвольным доступом.

Для *описания переменной* определенного типа используется инструкция ***Dim***, имеющая вид:

Dim **ИмяПеременной** ***As*** **ТипДанных**

(при описании синтаксических правил полужирным шрифтом показаны ключевые слова VBA, а курсивом – синтаксические обозначения, которые при вводе текстов программ должны заменяться на конкретные значения, имена и т.п.). Например:

Dim **X** ***As Integer***

Эта инструкция описывает переменную **X** как целое число типа Integer (с диапазоном значений от -32768 и до 32767). При попытке присвоить **X** значение, выходящее за пределы этого диа-

пазона, возникает ошибка. При присваивании **X** дробного числа, выполняется округление. Например:

```
X = 32768    ' Вызывает ошибку.
X = 5.9      ' Задает для X значение 6.
```

Таким образом, явное указание типов переменных является средством предотвращения ошибок, неправильного использования переменных в программах.

В одном операторе ***Dim*** может быть описано несколько переменных, тогда их описания перечисляются через запятую:

```
Dim ИмяПеременной As ТипДанных, ..., _
    ИмяПеременной As ТипДанных
```

В зависимости от того, где и как описана переменная, определяется область ее видимости и время жизни.

Временем жизни (областью действия) переменной называется время, в течение которого переменная доступна и может иметь значение. Значение переменной может меняться на протяжении ее времени жизни, но в течение этого времени она обязательно имеет какое-либо значение. Когда переменная теряет область определения, она более не имеет значения.

Переменная имеет область действия уровня процедуры, если она описана в теле подпрограммы.

В начале выполнения процедуры в VBA все переменные инициализируются. Например: числовая переменная получает значение 0, строка переменной длины получает значение пустой строки (""), а строка фиксированной длины заполняется ASCII-символом 0 или ***Chr***(0). Переменные типа Variant получают при инициализации значение ***Empty***. Каждый элемент массива переменных с определяемым пользователем типом при инициализации получает значение, которое он получил бы, если бы являлся одиночной переменной.

Если описание переменной помещено в раздел описаний модуля, перед описаниями всех процедур и функций, переменная имеет область действия уровня модуля. Такие переменные доступны во всех процедурах модуля.

Переменные с областью действия уровня проекта описываются с использованием ключевого слова ***Public***. Описания таких переменных размещаются в разделе описания любого модуля проекта.

Описание с ключевым словом ***Static*** переменных в процедурах позволяет «жить» таким переменным до окончания выполнения кода, т.е. их значения сохраняются между вызовами процедуры.

При описании объектной переменной для нее выделяется память, но ее значение определяется как Nothing до тех пор, пока ей не присвоена ссылка на объект с помощью инструкции ***Set***.

Если значение переменной не изменяется во время выполнения программы, она сохраняет значение, полученное при инициализации, до тех пор, пока не потеряет область определения.

Переменная, описанная с помощью инструкции ***Dim*** *на уровне процедуры*, сохраняет значение до окончания выполнения процедуры. Если процедура вызывает другие процедуры, переменная сохраняет свое значение, пока не закончится выполнение и этих процедур.

Если переменная уровня процедуры описана с помощью ключевого слова ***Static***, она сохраняет свое значение до тех пор, пока программа выполняется в каком-либо модуле (т.е. такая переменная является *статической*). По завершении работы всей программы переменная теряет свою область определения и свое значение. Ее время жизни совпадает со временем жизни переменной уровня модуля.

Переменная уровня модуля отличается от статической переменной. В стандартном модуле или в модуле класса она сохраняет свое значение до завершения выполнения программы. В модуле класса она сохраняет значение, пока существует экземпляр этого класса. Переменные уровня модуля занимают память, пока их значения не удалены, поэтому их следует использовать только при необходимости.

Если перед инструкциями ***Sub*** или ***Function*** в заголовке процедуры имеется ключевое слово ***Static***, значения всех переменных уровня процедуры сохраняются между вызовами процедуры.

В VBA, если переменная не описана, по умолчанию для нее используется тип Variant, но тип данных по умолчанию можно изменить, поместив инструкции описания ***Def*Тип** в начало модуля. Этот оператор использует в качестве аргументов начальные буквы имен переменных, которым по умолчанию будет припи-

сан заданный в операторе тип. Данная инструкция начинается с ключевого слова, обозначающего тип (например: ***DefInt*** – для определения начальных букв имен целочисленных переменных, ***DefStr*** – строковых переменных). В одной строке может быть несколько инструкций, отделенных друг от друга двоеточием:

DefStr **S:** ***DefInt*** **I:** ***DefCur*** **C**

После такой декларации все неописанные переменные, имена которых начинаются на **I**, будут отнесены к типу ***Integer***, на **C** – к типу ***Currency***, на **S** – к ***String***.

VBA позволяет разработчику создавать свои комбинированные типы. При использовании таких типов в одной переменной можно группировать несколько значений различных типов. Пользовательский тип определяется инструкцией ***Type***. Это аналог структур или записей в других языках.

Дополнительные сведения о присвоения имен переменным содержатся в разделе справочника Visual Basic «Правила присвоения имен в языке Visual Basic».

6.2.6. Объекты в программах на VBA и объектные переменные

Код VBA большей частью основан на использовании *объектов*. Например, презентация PowerPoint, рабочая книга Excel или диапазон ячеек являются объектами. Объекты – это элементы, которыми можно управлять. Объекты Microsoft Office образуют иерархии (они более подробно рассматриваются ниже, при изучении конкретных приложений). Все объекты документированы во встроенной справочной системе VBA.

Каждый объект имеет *свойства* (атрибуты, характеризующие его) и *методы* (операции, которые можно выполнять над данным объектом). При обращении к конкретному свойству или методу объекта необходимо указать объект и нужное свойство или метод. Общий синтаксис, используемый при манипулировании объектами следующий:

Объект.Свойство или **Объект.Метод**

В одной строке кода можно выполнить только один метод, вызвав его:

Объект.Метод ()

Для вызова метода необходимо указать объект и метод. Кроме того, большинство методов имеют набор аргументов, позволяющих выполнить дополнительную настройку. Аргументы могут быть обязательными и необязательными. Если при вызове метода необязательные аргументы не передаются, они принимают значения по умолчанию. Аргументы при вызове метода могут передаваться по имени и в соответствии с позицией, как и аргументы процедур.

Одна строка кода может содержать оператор присваивания, устанавливающий свойство объекта (задающий значение свойства объекта, записанного в левой части оператора присваивания, равным значению выражения, записанного в правой части оператора присваивания):

Объект.Свойство = Выражение

Значение свойства объекта может быть получено и использовано при вычислении значения выражения.

Некоторые объекты имеют свойство «по умолчанию» – свойство, к которому происходит обращение, если в операторе указан только объект.

При записи полного имени объекта в нем перечисляются имена всех объектов, находящихся на предшествующих уровнях иерархии (например, для указания диапазона ячеек на рабочем листе книги Excel нужно указать имя рабочей книги, имя рабочего листа и имя или координаты диапазона). Эти имена записываются через точку.

Объектные переменные могут хранить ссылки на объекты. Их целесообразно использовать, если обращение к объекту производится многократно. Такие переменные создаются в два шага: сначала ее нужно объявить, а затем установить значение. Одну и ту же переменную можно использовать для ссылки на различные объекты. Но это снижает эффективность программы, так как при использовании типа данных ***Object*** происходит позднее (во время выполнения программы) связывание. Раннее связывание происходит во время компиляции. Для его реализации нужно объявлять переменные, указывая в качестве типа конкретный *класс объекта* (например, ***Range*** или ***Presentation***), т.е.

классы для объектов играют ту же роль, что и типы для переменных.

В программах на VBA можно работать как с единичными объектами, так и с семействами объектов. *Семейство* определяется как группа подобных объектов. Семейство содержит упорядоченное множество элементов (например, множество элементов управления данной формы или множество сценариев).

На единичные объекты ссылаются непосредственно по имени, а *на объект в семействе ссылаются по индексу в семействе* (например, по номеру можно сослаться на рабочий лист). Индексы объектов в семействах всегда начинаются с 1. При обращении по номеру (индексу) следует учитывать, что нумерация осуществляется в том порядке, в котором объекты добавлялись в семейство. Таким образом, к любому объекту в семействе можно обратиться как по его собственному имени, так и по индексу в семействе. Способ обращения зависит от решаемой задачи (если нужно перебрать все элементы семейства или какое-то их подмножество, удобнее работать с индексами; если же выполняется операция над конкретным объектом, лучше использовать его имя).

Семейству принадлежат не уникальные объекты, а объекты, которые могут существовать в нескольких экземплярах в каждом контексте. Хотя семейство содержит группы объектов, само семейство тоже является объектом. Семейство также имеет связанные с ним свойства и методы. VBA предоставляет три основных метода для работы с семействами: ***Add*** (добавить элемент в семейство), ***Remove*** (удалить элемент из семейства), ***Count*** (количество элементов в семействе).

Когда приложение используется для управления объектами из другого приложения, необходимо создать ссылку на библиотеку типов второго приложения. Когда ссылка определена, имеется возможность описать объектные переменные с наиболее подходящим для них типом. Например, если при работе в Microsoft Access определяется ссылка на библиотеку типов Microsoft Excel, то внутри Microsoft Access можно описать переменную типа Worksheet, чтобы она представляла объект Worksheet Microsoft Excel.

Если для управления объектами Microsoft Access используется другое приложение, то, как правило, объектные переменные описываются с наиболее подходящим для них типом. Возможно также использование ключевого слова ***New*** для автоматического создания нового экземпляра объекта. Однако необходимо указать, что это объект Microsoft Access. Например, если описывается объектная переменная, представляющая форму Microsoft Access внутри Microsoft Visual Basic, необходимо различать объект Form Microsoft Access и объект Form Visual Basic. Имя библиотеки типов включается в описание переменной, как показано в следующем примере:

```
Dim frmOrders As New Access.Form
```

Некоторые приложения не распознают отдельные объектные типы Microsoft Access. Даже если в этих приложениях создана ссылка на библиотеку типов Microsoft Access, необходимо описать все объектные переменные Microsoft Access с типом ***Object***. В этом случае ключевое слово ***New*** также не может использоваться для создания нового экземпляра объекта. В следующем примере показано, как в таком приложении надо описывать переменную, чтобы она представляла экземпляр объекта Application Microsoft Access; затем приложение создает экземпляр объекта Application.

```
Dim appAccess As Object
Set appAccess = CreateObject("Access.Application")
```

Синтаксис, поддерживаемый конкретным приложением, описан в документации по приложению.

Для подключения ссылок можно использовать команду меню **Сервис**.

6.2.7. Создание объектов пользователя

В VBA у пользователя имеется возможность создавать собственные объекты. *Модули класса* содержат информацию о создаваемых пользователем объектах. Модуль класса содержит код, используемый для создания объектов, задания их свойств, код методов объектов. Имя соответствующего объекта (точнее *класса* объектов) используется в качестве имени модуля класса.

При описании свойств и методов нового объекта использу-

ется специальный синтаксис.

В объектно-ориентированном программировании используется понятие абстракции. *Абстракция* – выделение только тех свойств, которые существенны, интересны на определенном уровне рассмотрения. Классы определяют информацию об объекте, действия, выполняемые объектом и над объектом. *Класс* – это «проект» объектов. Объекты, созданные на основе одного и того же класса, обладают одинаковым набором свойств и методов, но значения свойств могут быть разными, они задаются при порождении экземпляра объекта определенного класса. *Экземпляром класса* называется объект, созданный на основе класса. На основе одного и того же класса можно создать произвольное количество экземпляров.

Для создания собственного класса необходимо выполнить команду **Вставка ▸ Модуль класса**. После выполнения этой команды в окне проекта у текущего проекта появляется ветвь, представляющая новый класс, и открывается окно модуля этого класса. Для ввода имени нового класса вместо присвоенного ему при создании стандартного имени нужно открыть окно свойств этого компонента и изменить соответствующее свойство (***Name***). Других свойств у класса пока нет.

При создании свойств класса требуется предусмотреть возможность выполнения двух операций: получения текущего значения свойства и установки значения свойства. Эти операции выполняются с помощью процедур ***Property Get*** и ***Property Let***. В заголовках этих процедур указывается имя соответствующего свойства.

Процедура ***Property Let*** используется для установки значения свойства. Основной функцией этой процедуры является проверка того, является ли устанавливаемое значение допустимым. Устанавливаемое значение всегда задается в инструкции ***Property Let*** (заголовке процедуры) последним параметром. Переданное при обращении к процедуре значение аргумента, если оно соответствует условию допустимости, становится текущим значением свойства и запоминается в переменной уровня модуля, доступной в любой процедуре модуля класса.

Процедура ***Property Get*** используется для получения текущего значения свойства. Данная процедура просто должна присвоить значение соответствующей переменной уровня модуля свойству (т.е. эта процедура должна содержать оператор присваивания, в левой части которого указывается имя свойства).

Свойства можно создать автоматически с помощью диалогового окна команды **Вставка ▸ Процедура**. В поле «Имя» диалогового окна «Вставка процедуры» задается имя свойства. Переключатель группы «Тип» устанавливается в позиции «Свойство». Если свойство имеет общую область определения, т.е. доступно для просмотра и изменения программисту, использующему объект, то следует установить в группе «Область определения» переключатель «Общая», если же свойство можно использовать только в модуле класса, то выбирается переключатель «Личная». Скрытые, недоступные для использования программистом компоненты можно просмотреть с помощью команды **Показать скрытые компоненты** контекстного меню окна просмотра объектов.

В результате выполнения программы в модуль класса вставляются «заготовки» процедур. Например, если с объектом связан файл, имя этого файла является свойством этого объекта. Описания переменной для хранения этого свойства при работе с объектом и процедур установки/получения значения свойства могут быть следующими:

```
Dim Pic_File As String   ' Имя файла объекта
Public Property Get Файл() As String
' Свойство, указывающее файл, в котором хранится объект
      Файл = Pic_File
End Property
Public Property Let Файл(ByVal vNewValue As String)
' Установка свойства - файла, в котором хранится объект.
' Файл может размещаться только на диске C
      If (vNewValue <> "") And _
            (Left(vNewValue, 3) = "C:\") And _
            (Len(vNewValue) > 3) Then
               Pic_File = vNewValue
      Else
            x = MsgBox("Недопустимое имя файла!", _
                     vbCritical, "Ошибка в свойстве")
      End If
End Property
```

Для создания экземпляров объектов класса и работы с ними нужно описать объектные переменные соответствующих типов инструкциями вида

```
Dim ОбъектнаяПеременная As Класс
```

и присвоить им значения с помощью инструкции

```
Set ОбъектнаяПеременная = New Класс
```

Предположим, что в проекте создан модуль класса «МойКласс», в который включены приведенные выше инструкции. Свойство «Файл» доступно программно, в приведенной ниже процедуре создается объект, принадлежащий классу «МойКласс», устанавливается его свойство «Файл» и значение свойства выводится на экран:

```
Public Sub СозданиеОбъектаПользователя()
   Dim A As МойКласс
   Set A = New МойКласс
   MsgBox A.Файл
   A.Файл = "C:\Объект"
   MsgBox A.Файл
End Sub
```

Код процедур установки и получения свойства можно ввести вручную в окне программы модуля класса.

Методы используются для реализации операций над объектом, изменения его свойств в соответствии с определенными правилами. Создание методов класса – это написание процедур. Если метод вычисляет и возвращает некоторое значение, то необходимо написать функцию. Если метод выполняет определенные действия, но не возвращает результат, пишется подпрограмма.

Если в окне модуля класса выбрать из списка объектов строку «Class», в списке «Процедура» появляются два элемента. С их помощью можно создать процедуру инициализации (***Initialize***) и уничтожения (***Terminate***) для объектов этого класса. В процедуре инициализации можно записать код для присваивания начальных значений свойствам создаваемого объекта. А в процедуре уничтожения выполнить действия, связанные с удалением этого объекта. Например, в модуль класса «МойКласс» можно вставить код инициализации объекта:


```
Private Sub Class_Initialize()
   Pic_File = "C:\Temp\ObjFile"
   File_Number = FreeFile
   Open Pic_File For Random As File_Number
End Sub
```

где переменная **File_Number** описана на уровне модуля:

```
Dim File_Number As Integer ' Номер файла
```

При создании экземпляра объекта данного класса открывается или создается файл.

Если при работе с объектом используются, например, какие-либо временные файлы, то при уничтожении объекта можно удалить эти файлы:

```
Private Sub Class_Terminate()
   MsgBox "Уничтожение!"
   Close File_Number
   Kill (Pic_File)
End Sub
```

При уничтожении объекта, если выполнить приведенную выше процедуру СозданиеОбъектаПользователя, произойдет ошибка, так как в ней изменено свойство, определяющее имя файла.

В модуле класса можно описать специальный метод создания объектов с определенными свойствами, задаваемыми параметрами метода.

6.2.8. Описание и использование констант

Кроме переменных, в программах на VBA используются и *константы*, значения которых являются постоянными и не изменяются в ходе выполнения программы.

В VBA существуют так называемые *встроенные константы*. Это константы, имена которых определены в самом языке. Все встроенные константы VBA (их более 250) имеют имена вида ***vb*ИмяКонстанты**, встроенные константы Access имеют имена вида ***ac*ИмяКонстанты**, встроенные константы Excel – ***xl*ИмяКонстанты**, Word – ***wd*ИмяКонстанты**, PowerPoint –

***pp*ИмяКонстанты**. Значения встроенных констант можно просмотреть в окне «Просмотр объектов», которое открывается при нажатии клавиши *F2* в Редакторе Visual Basic. Для получения информации о конкретной константе можно, установив курсор на ее имени в окне программы, нажать клавишу *F1*.

При описании константы ей можно присвоить значащее имя. Инструкция ***Const*** используется для описания константы и определения ее значения. После описания константу нельзя модифицировать и нельзя присваивать ей новое значение.

Константы имеют те же три уровня областей действия, что и переменные, но способ описания констант каждого уровня отличается от способа описания переменных.

Константа описывается в процедуре или в начале модуля, в разделе описаний.

Константы уровня процедуры доступны только внутри процедуры, в которой они описаны.

Константы уровня модуля описываются в разделе описания модуля. Они доступны в любой процедуре модуля, где они описаны. Константы уровня модуля по умолчанию являются личными, поэтому ключевое слово ***Private*** в описании можно не использовать. При описании общих констант уровня проекта инструкции ***Const*** должно предшествовать ключевое слово ***Public***. Такие константы можно использовать в любом месте проекта, в котором они описаны.

Дополнительные сведения содержатся в разделе «Область определения и видимость» справочника Visual Basic.

Константы могут быть описаны одним из следующих типов данных: ***Boolean***, ***Byte***, ***Integer***, ***Long***, ***Currency***, ***Single***, ***Double***, ***Date***, ***String***, или ***Variant***. Поскольку значение константы уже известно, можно задать тип данных в инструкции ***Const***. Дополнительные сведения содержатся в разделе «Типы данных» справочника Visual Basic.

Допускается также описание нескольких констант в одной строке. В этом случае, чтобы задать тип данных, надо указать определенный тип для каждой константы. В следующем примере описываются две константы:

```
Const cAge As Integer = 34, cWage As Currency = 35000
```


6.2.9. Массивы в VBA

Массивы являются мощным средством обработки больших объемов данных.

Массив – то индексированная группа значений одного и того же типа. Одномерный массив – это индексированный список значений. Двумерный массив можно представить как таблицу, а трехмерный – как группу таблиц, каждая из которых имеет одно и то же количество строк и столбцов. Массивы VBA могут иметь до 60 измерений.

Как и другие переменные, массивы описываются с помощью инструкций ***Dim***, ***Static***, ***Private*** или ***Public***. Разница между скалярными переменными и массивами состоит в том, что для последних надо указывать размер массива. Массив с заданным размером называется массивом *фиксированного размера*. Массив с переменным размером называется *динамическим*.

Размерность массива фиксированного размера и количество элементов в нем указывается в круглых скобках сразу за именем массива в его описании. Каждому измерению соответствует одно число, задающее верхнюю границу соответствующего измерения. Начало индексации массива (с 0 или 1) определяется параметрами инструкции ***Option Base***. Если не указано ***Option Base*** 1, нижняя граница индексов массива равняется нулю. В следующей строке программы массив фиксированного размера описывается как массив типа ***Integer***, имеющий 11 строк и 11 столбцов:

```
Dim MyArray (10, 10) As Integer
```

Первый аргумент представляет строки, а второй – столбцы.

Можно задать индексацию массива с любого выбранного числа. Для этого при описании массива задаются и верхняя, и нижняя границы индекса:

```
Dim DataArray (5 to 10) As Byte
Dim RealMulti(1 To 5, 1 To 10) As Single
```

В этом примере массив **DataArray** – это одномерный массив, содержащий шесть элементов (чисел типа ***Byte***), а **RealMulti** – двумерный массив вещественных чисел из 5 строк и 10 столбцов.

Как и при описании других переменных, если тип данных при описании массива не задается, подразумевается, что элемен-

ты массива имеют тип ***Variant***.

Динамические массивы используются, когда известно, что размер элемента может меняться в ходе выполнения программы. Если массив описан как динамический, можно изменять его размер во время работы программы, соизмеряя его с текущими потребностями. Для описания динамического массива используются инструкции ***Dim***, ***Static***, ***Private*** или ***Public*** с пустыми скобками, как показано в следующем примере:

```
Dim DynArray() As Single
```

Для переопределения массива используется оператор ***ReDim***. После этого массиву выделяется необходимая для его размещения память. Инструкция ***ReDim*** используется в процедуре внутри области определения массива для изменения числа размерностей, определения числа элементов и задания верхних и нижних границ индексов для каждой размерности. Инструкцию ***ReDim*** можно применять для изменения динамического массива столько раз, сколько потребуется. Однако при каждом применении данные, содержащиеся в массиве, теряются. Инструкция ***ReDim Preserve*** увеличивает размер массива, сохраняя при этом его содержимое. В следующем примере показывается, как можно увеличить массив **varArray** на 10 элементов без уничтожения текущих значений элементов массива:

```
ReDim Preserve VarArray(UBound(VarArray) + 10)
```

Использование ключевого слова ***Preserve*** вместе с динамическим массивом позволяет изменить только верхнюю границу последней размерности массива, изменение числа размерностей невозможно.

Размер массива можно и уменьшить. Если уменьшить размер массива, данные из удаленных элементов будут потеряны.

При передаче массива в процедуру по ссылке нельзя изменять размеры массива в процедуре.

При необходимости сослаться на все элементы массива можно ссылаться на массив как целое. Возможны также ссылки на его отдельные элементы. Чтобы задать значение отдельного элемента массива, надо указать в круглых скобках за именем массива индекс нужного элемента.

Для работы с массивами чаще всего используются функции

Array, ***Erase***, ***IsArray***, ***Lbound***, ***Ubound***. Функция ***Array*** позволяет создавать массивы во время выполнения программы. Функция ***Erase*** используется для очистки массива. ***Erase*** повторно инициализирует элементы массивов фиксированной длины и освобождает память, отведенную для динамического массива, поэтому для работы с динамическим массивом его нужно переопределить с помощью ***ReDim***. Встроенная функция ***IsArray*** проверяет, является ли переменная массивом. Функции ***Lbound*** и ***Ubound*** позволяют определить верхнюю и нижнюю границы индексов.

При использовании типа ***Variant*** элементы массива могут иметь различный тип, например:

```
Dim VarData(3) As Variant
varData(0) = "Мария Петрова"
varData(1) = "Зеленая, 19"
varData(2) = 38
varData(3) = Format ("06-09-1952", "General Date")
```

Тот же результат может быть получен при использовании функции ***Array***:

```
Dim VarData As Variant
VarData = Array("Иван Петров", "Зеленая, 19", 38, _
                Format("06-09-1952", "General Date"))
```

Независимо от способа создания массива значений ***Variant*** его элементы нумеруются индексами. Например, следующая инструкция может быть добавлена к любому из предыдущих примеров.

```
MsgBox "Записаны данные для " & varData(0) & "."
```

В справочной системе VBA можно получить более подробную информацию о работе с массивами.

6.2.10. Описание типов данных пользователя

На основе стандартных типов данных VBA пользователь может описать собственные структурированные типы данных. Переменные таких типов могут содержать разнородную информацию.

Тип описывается с помощью инструкции ***Type***:

```
[Private | Public] Type ИмяТипа
   ИмяЭлемента [([Индексы])] As Тип
```

```
    [ИмяЭлемента [([Индексы])] As Тип]
    ...
End Type
```

В описании может быть указано произвольное количество элементов любого типа, включая массивы и другие заданные пользователем типы. Новый тип – аналог записей или структур в других языках. Переменные созданных с пользователем типов можно описать с помощью инструкции вида

```
Dim ИмяПеременной As ИмяТипа
```

После описания переменной ее элементам можно присвоить значения. Для доступа к элементу нужно указать имя переменной и имя элемента, заданное в описании типа (разделителем является точка). Например, оператор присваивания значения будет иметь вид

```
ИмяПеременной.ИмяЭлемента = Значение
```

На основе описанных пользователем типов можно создавать массивы и новые типы.

6.2.11. Запись выражений на VBA

Выражение – часть кода, при выполнении которой вычисляется некоторое значение. При вычислении могут быть получены значения различных типов. Тип результата зависит от типов операндов и операций, которые над ними выполняются.

Основные операции, реализуемые в программах на VBA, делятся на группы:

– *арифметические операции*:
  · сложение (+),
  · вычитание (–),
  · умножение (*),
  · деление (/), дающее в качестве результата число с плавающей точкой,
  · деление (\), возвращающее целочисленный результат – частное от деления,
  · ***Mod*** – получение остатка от деления целых чисел,
  · возведение в степень (^),

– *логические операции*:
  · ***Not*** – отрицание,
  · ***And*** – конъюнкция (логическое «И»),
  · ***Or*** – дизъюнкция (логическое «ИЛИ»),
  · ***Eqv*** – логическая эквивалентность («равенство»),
  · ***Imp*** – импликация,
  · ***Xor*** – исключающее «ИЛИ»,
– *операции сравнения*:
  · равно (=),
  · не равно (<>),
  · меньше (<),
  · больше (>),
  · меньше или равно (<=),
  · больше или равно (>=),
– *операция конкатенации* (**&**) – «склеивание» строк.

Если выражение содержит несколько операций (в справочном руководстве по VBA их называют операторами), то значения «подвыражений» рассчитываются в определенном порядке. Такой порядок называют *порядком старшинства* или *приоритетом* операций. Арифметические и логические операции выполняются в порядке их расположения в следующей табл. 6.1.

Таблица 6.1. **Приоритеты операций**

Арифметические	Логические
Возведение в степень (^)	***Not***
Изменение знака (–)	***And***
Умножение и деление (*, /)	***Or***
Целое деление (\)	***Xor***
Деление по модулю (***Mod***)	***Eqv***
Сложение и вычитание (+, –)	***Imp***
Слияние строк (**&**)	

Если выражение содержит операторы разных типов, то первыми выполняются арифметические операции, следом за ними операции сравнения, а последними логические операции. Все операции сравнения имеют равный приоритет, т.е. выполняются в порядке их расположения в выражении слева направо.

Стоящие рядом в выражении операции умножения и деления выполняются слева направо. В таком же порядке выполняются стоящие рядом операторы сложения и вычитания. Операции внутри круглых скобок всегда выполняются раньше, чем операции вне скобок.

Операция конкатенации (**&**) по порядку старшинства следует сразу за арифметическими операциями и перед операциями сравнения.

Операция ***Like***, равная по старшинству остальным операциям сравнения, выделяется в самостоятельный тип операций сравнения с образцом.

Операция ***Is*** является операцией сравнения ссылок на объект. Эта операция не выполняет сравнение объектов или их значений – она проверяет только, указывают ли две разные ссылки на один объект.

6.2.12. Рекурсивные вычисления

VBA позволяет организовать рекурсивные вычисления, создавать рекурсивные процедуры, т.е. процедуры, которые явно или косвенно вызывают себя. При этом следует помнить, что код рекурсивной процедуры должен содержать проверку условия завершения рекурсивных вызовов. Каждый новый вызов рекурсивной процедуры требует выделения памяти для ее переменных, поэтому следует подумать, а нельзя ли заменить рекурсию итерацией, вложенными циклами, если есть проблема нехватки памяти.

При описании процедур-функций приведен пример рекурсивного вычисления факториала.

6.2.13. Процедуры-подпрограммы

Подпрограмма представляет собой последовательность операторов языка Visual Basic, ограниченных инструкциями ***Sub*** и ***End Sub***. Подпрограмма выполняет действия, заданные перечисленными в ней инструкциями, а результаты работы может вернуть, если это необходимо, через параметры, переданные ей при вызове по ссылке. Таким образом, эти процедуры нельзя вызывать при вычислении значений выражений, вызов подпрограмм

осуществляется отдельными операторами (инструкциями). Если подпрограмма не имеет аргументов, инструкция ***Sub*** (заголовок процедуры) должна содержать пустые скобки вслед за именем, назначенным процедуре.

Правило записи процедуры-подпрограммы следующее:

```
Sub ИмяПодпрограммы (СписокФормальныхПараметров)
        КодПодпрограммы
End Sub
```

В списке параметров перечисляются через запятую описания аргументов, передаваемых подпрограмме. Для каждого параметра указывается его имя (обязательно), способ передачи в подпрограмму, тип, а также является ли аргумент обязательным при вызове и следует ли упаковывать следующие аргументы в массив. Используемые в описании списка формальных параметров ключевые слова и их назначение описаны выше. Далее будут приведены примеры описаний подпрограмм. Перед ключевым словом ***Sub*** могут быть указаны ключевые слова ***Public*** / ***Private***, ***Static***.

Чтобы вызвать процедуру-подпрограмму из другой процедуры, следует указать имя этой процедуры и значения для всех требуемых аргументов. Использование инструкции ***Call*** не обязательно, однако, если она все же используется, аргументы должны быть заключены в скобки.

Выйти из подпрограммы можно «досрочно», не через конец, с помощью инструкции ***Exit Sub***.

В следующем примере инструкция ***Sub*** (с парной ей инструкцией ***End Sub***) описывает процедуру с именем **ApplyFormat**. Все инструкции, заключенные между ***Sub*** и ***End Sub*** выполняются всегда, когда вызывается или выполняется процедура **ApplyFormat**:

```
Sub ApplyFormat()
     Const Limit As Integer = 33
     Dim MyCell As Range

     ' Другие инструкции …

End Sub
```

В описанную выше процедуру при обращении к ней не передаются при вызове параметры.

В следующем примере процедура **Main** вызывает процедуру **MultiBeep**, передавая значение 56 для ее аргумента. По окончании работы **MultiBeep** управление возвращается к **Main** и **Main** вызывает процедуру **Message**. Message показывает окно сообщения, когда пользователь выбирает мышью **OK**, управление возвращается к **Main** и **Main** завершается.

```
Sub Main()
   MultiBeep 56
   Message
End Sub

Sub MultiBeep(numbeeps)
   For counter = 1 To numbeeps
      Beep ' Передача звукового сигнала через динамик
   Next counter
End Sub

Sub Message()
   MsgBox "Пора сделать перерыв!"
End Sub
```

Аргументы могут передаваться в подпрограмму как по ссылке, так и по значению. Пример, показывающий разницу в полученных результатах при использовании различных способов передачи параметров, приведен ниже, при описании функций.

Для передачи аргументов в процедуру может использоваться массив параметров. При описании процедуры не требуется указывать число элементов такого массива.

Для обозначения массива параметров используется ключевое слово ***ParamArray***. Такой массив описывается как массив типа Variant и всегда представляет последние элементы из списка аргументов в описании процедуры. Этот способ передачи параметров может использоваться, когда может изменяться количество передаваемых в процедуру параметров. Ниже приводится пример описания процедуры с массивом параметров:

```
Sub AnyNumberArgs  (strName As String, _
                    ParamArray intScores() As Variant)
   Dim intI As Integer
```



```
   Debug.Print strName; "    Дети:"
   For intI = 0 To UBound(intScores())
      Debug.Print "          "; intScores(intI)
   Next intI
End Sub
```

В следующих строках приведены примеры вызова этой процедуры с различным числом аргументов:

```
AnyNumberArgs "Иванов И.И.", "Саша", "Даша"
AnyNumberArgs "Петрова П.П.", "Ася", "Петя", "Вася", "Коля"
```

При вызове процедур ***Sub*** или ***Function*** возможна *позиционная передача аргументов*, т.е. в порядке следования в описании процедуры. Кроме того, аргументы могут передаваться *по именам*, вне зависимости от позиции. Именованный аргумент состоит из имени соответствующего формального параметра, за которым следует двоеточие со знаком равенства (**:=**) и значение передаваемого фактического параметра.

Именованные аргументы особенно полезны при вызове процедуры с необязательными аргументами (***Optional***). Если используются именованные аргументы, то запятые для обозначения отсутствующих параметров не нужны. С помощью именованных аргументов проще проследить, какие аргументы переданы, а какие опущены.

В описании процедуры перед необязательными аргументами должно стоять ключевое слово ***Optional***. Кроме того, в описании процедуры можно присвоить значение необязательному аргументу, которое он принимает по умолчанию. Если при вызове процедуры с аргументом ***Optional*** этот аргумент не описан, то для него используется имеющееся значение по умолчанию. Если по умолчанию значение не присвоено, аргумент обрабатывается так же, как любая другая переменная указанного типа.

В следующей процедуре имеются необязательные параметры. Функция **IsMissing** определяет, были ли переданы в процедуру необязательные аргументы.

```
Sub OptionalArgs  (strState As String, _
                   Optional intRegion As Integer, _
                   Optional strCountry As String = "USA")
   If IsMissing(intRegion) And IsMissing(strCountry) Then
      Debug.Print strState
   ElseIf  IsMissing(strCountry) Then
```

```
    Debug.Print strState, intRegion
  ElseIf  IsMissing(intRegion) Then
    Debug.Print strState, strCountry
  Else
    Debug.Print strState, intRegion, strCountry
  End If
End Sub
```

В эту процедуру можно передать, например, именованные аргументы:

```
OptionalArgs strCountry:="USA", strState:="MD"
OptionalArgs strState:= "MD", intRegion:=5
```

При передаче аргументов по занимаемой ими позиции на месте пропущенных параметров ставится обозначающая их позицию запятая.

6.2.14. Процедуры-функции

Функция представляет собой последовательность инструкций языка Visual Basic, ограниченных инструкциями ***Function*** и ***End Function***. Эти инструкции выполняются при вызове функции. При обращении к функции ей можно передать параметры. Если процедура ***Function*** не имеет аргументов, ее заголовок должен содержать пустые скобки за именем функции. Процедура ***Function*** возвращает вычисленное при выполнении записанных в ней инструкций значение, следовательно, функцию можно вызвать в выражении, возвращенный из нее результат может быть использован в качестве операнда при вычислении значения выражения. Возврат значения осуществляется путем его присвоения имени функции в одной или нескольких инструкциях процедуры.

```
Function ИмяФункции (СписокФормальныхПараметров) As Тип
  КодФункции
  ИмяФункции = Результат
End Function
```

Название типа, указанное в заголовке, определяет тип результата, возвращаемого функцией. Это указание можно опустить. Перед ключевым словом ***Function*** могут быть указаны ключевые слова ***Public*** / ***Private***, ***Static***. При вызове функции указывается ее имя, используются те же способы передачи параметров, что и при вызове подпрограммы. Инструкция ***Exit Function*** позволяет выйти из функции «досрочно», но следует помнить, что предварительно должен быть определен результат.

В следующем примере функция **Celsius** пересчитывает градусы Фаренгейта в градусы Цельсия. Когда функция вызывается процедурой Main, переменная, содержащая значение аргумента, передается функции. Результат вычислений возвращается вызывающей процедуре и выводится в окно сообщения.

```
Sub Main()
 temp =  Application.InputBox(Prompt:= _
      "Введите температуру в градусах Фаренгейта.", _
      Type:=1)
 MsgBox  "Температура равна " & Celsius(temp) _
         & " градусов Цельсия."
End Sub

Function Celsius(fDegrees)
    Celsius = (fDegrees - 32) * 5 / 9
End Function
```

Приведенная ниже функция демонстрирует пример рекурсивных вычислений.

```
Function Factorial (ByVal MyVar As Integer) As Integer
  MyVar = MyVar - 1
  If  MyVar = 0 Then
    Factorial = 1
    Exit Function
  End If
  Factorial = Factorial (MyVar) * (MyVar + 1)
End Function
```

Эту функцию можно вызвать, передав в качестве параметра, например, переменную:

```
S = 5
Print S ' Выводит на экран 5
Print Factorial(S) ' Выводит на экран 120=5!
Print S ' Выводит на экран 5
```

Аргумент передается в функцию по значению. Без ключевого слова ***ByVal*** в описании функции приведенные выше инструкции ***Print*** выводят на экран 5, 1 и 0. Причина заключается в том, что в этом случае аргумент передается по ссылке, т.е. **MyVar**

ссылается на переменную **S**, которая уменьшается, следовательно, на 1 до тех пор, пока не станет равной 0.

6.2.15. Процедуры-свойства

Процедура ***Property*** представляет собой последовательность инструкций языка Visual Basic, которые позволяют программисту создавать собственные свойства объектов и оперировать с ними:

- Процедуры ***Property*** создают допускающие только чтение свойства для форм, стандартных модулей и модулей класса.
- Процедуры ***Property*** следует использовать вместо переменных ***Public*** в программе, которая должна выполняться после задания значения свойства.
- В отличие от переменных ***Public*** процедуры ***Property*** могут иметь справочные строки, присвоенные им в окне «Просмотр объектов».

Когда создается процедура ***Property***, она становится свойством модуля, содержащего эту процедуру. В языке Visual Basic имеются три типа процедур ***Property***:

Процедура	Описание
Property Let	Присваивает значение свойству
Property Get	Возвращает значение свойства
Property Set	Задает ссылку на объект

Синтаксис процедуры ***Property***:

```
Property  {Get | Let | Set} ИмяСвойства _
          (СписокАргументов) As Тип
   КодПроцедуры
End Property
```

Процедуры ***Property*** обычно используются попарно: ***Property Let*** с ***Property Get*** и ***Property Set*** с ***Property Get***. Описание одной процедуры ***Property Get*** подобно описанию свойства, доступного только для чтения. Использование всех трех процедур ***Property*** вместе полезно только для переменных ***Variant***. В отличие от ***Property Let*** процедура ***Property Set*** предназначена для работы с объектами.

Более подробно информацию о процедурах-свойствах можно получить из справочного руководства по VBA.

6.2.16. Создание процедур

Модуль включает в себя процедуры, которые описывают выполнение приложением конкретных задач. Процедуры используются для разбиения сложных программистских задач на более простые элементы.

Разработчик имеет несколько возможностей для создания процедур.

Разработать процедуру можно просто путем ввода текста программы в редакторе VBA. Для этого нужно открыть модуль, для которого создается процедура. Затем можно ввести код процедуры в окне программы. Для добавления новой процедуры (подпрограммы или функции) следует установить курсор (точку вставки) за последней процедурой модуля, ввести ключевое слово (***Sub*** или ***Function***) и имя новой процедуры или функции и нажать клавишу *Enter*. Справку по синтаксису можно получить нажатием клавиши *F1*. Visual Basic автоматически завершит текст процедуры соответствующей инструкцией (***End Sub***, ***End Function*** или ***End Property***) сразу после ввода ее заголовка. Код процедуры вводится между заголовком и инструкцией ***End***.

Добавить процедуру в модуль можно также с помощью команды **Процедура…** меню **Вставка**. Имя процедуры и ее свойства можно будет ввести в открывшемся диалоговом окне «Вставка процедуры». В этом окне можно ввести имя процедуры, область ее действия и тип. Далее можно будет ввести код процедуры.

6.2.17. Управляющие структуры VBA и события

Процедуры VBA выполняются построчно от начала до конца, если последовательность их выполнения не будет изменена каким-либо событием или с помощью структуры управления.

Событие – это воздействующее на выполнение кода внешнее по отношению к нему действие (например: нажатие кнопки, истечение заданного промежутка времени и т.п.).

Управляющие структуры определяют порядок выполнения кода. Это инструмент для принятия решения в самой программе.

В VBA используются следующие структуры управления:

- ***If … Then … Else***
- ***Select … Case***
- ***For … Next***
- ***Do … Loop***
- ***While … WEnd***

Структура ветвления имеет несколько вариантов. Простейший синтаксис этого оператора следующий:

```
If  Условие Then
   Код
End If
```

Условие – любое выражение, результат вычисления которого имеет тип ***Boolean***. Условие может содержать операции сравнения и логические операции, в нем могут вызываться функции. При выполнении программы сначала проверяется указанное в операторе условие, а затем, если условие выполнено (результат вычисления выражения равен ***True***), исполняется код оператора. Если условие не выполнено (результат вычисления – ***False***), то код оператора пропускается.

Если при невыполнении условия также требуется выполнить какие-либо действия, используется более сложная форма оператора ветвления:

```
If  Условие Then
    КодИсполняющийсяПриВыполненииУсловия
Else
    КодИсполняющийсяПриНеВыполненииУсловия
End If
```

В зависимости от условия выполняется либо код, записанный в части ***Then***, либо код, содержащийся в части ***Else***.

При использовании структуры ветвления в ней можно проверить несколько вложенных условий:

```
If Условие1 Then
    Код1
ElseIf Условие2 Then
    Код2
  …
ElseIf УсловиеN Then
    КодN
Else
    КодElse
End If
```


Заданные в операторе условия проверяются последовательно, в программе будет выполнен код, соответствующий первому истинному условию, если же ни одно из условий не будет выполнено, управление передается на код в части ***Else***. Если часть ***Else*** отсутствует, то при невыполнении ни одного из условий, не выполняется ни одна из последовательностей операторов, код которых включен в оператор ветвления, – этот код пропускается.

Еще одна структура ветвления – оператор ***Select Case***. В данном операторе можно задать только одно выражение, управляющее ходом выполнения программы, значение которого должно быть вычислено и оценено:

```
Select Case КонтрольноеВыражение
    Case СписокВыражений1
         Код1
    Case СписокВыражений2
         Код2
    …
    Case СписокВыраженийN
         КодN
    Case Else
         КодElse
End Select
```

При выполнении данного оператора сначала вычисляется контролируемое выражение, а затем производится его сравнение с выражениями, заданными в списках вариантов ***Case***, и, если в каком-либо списке находится соответствующее значение, выполняется код, записанный в этом варианте ***Case***. Если соответствия нет (значение контрольного выражения не совпадает ни с одним из указанных в списках значений), выполняется код в части ***Case Else***. Если часть ***Case Else*** не задана, весь код оператора ***Select Case*** будет пропущен.

Структура повторения (цикл) повторяет заданные в нем строки кода, пока не получит команду выхода из цикла. Если такая команда в цикле не предусмотрена, цикл будет повторяться бесконечно. В VBA существует несколько видов циклов: циклы, выполняющиеся заданное количество раз, и циклы, повторяющиеся при выполнении или невыполнении заданных условий.

Оператор ***For … Next*** применяется, когда код должен повто-

ряться заданное количество раз. Эта структура управления имеет два варианта: цикл с параметром-счетчиком и цикл с параметром, принимающим все значения из заданной группы.

Цикл с параметром-счетчиком записывается по правилу:

```
For СчетчикЦикла = НачальноеЗначение _
    To КонечноеЗначение Step Шаг
       Код
Next СчетчикЦикла
```

Счетчик цикла управляет количеством повторений. Это целочисленная скалярная переменная. В начале выполнения цикла ей присваивается указанное начальное значение. Шаг задает разницу между двумя соседними значениями, которые принимает счетчик цикла: очередное значение счетчика цикла вычисляется как сумма текущего значения и заданного шага. Если значение шага не задано, по умолчанию шаг равен единице. Счетчик меняется, пока не достигнет конечного значения.

Код, включенный в цикл, повторяется для всех значений, принимаемых счетчиком. Указанные начальное и конечное значения и шаг задают условие повторения выполнения кода в цикле.

Шаг может быть как положительным, так и отрицательным. Его значение следующим образом определяет выполнение цикла: если значение положительно или 0, то цикл повторяется, пока значение счетчика цикла меньше или равно конечного значения; если шаг отрицательный, то выполнение кода в цикле будет повторяться, пока значение счетчика цикла больше или равно заданного конечного значения. Значение шага добавляется к значению переменной-счетчика после выполнения всех инструкций цикла. После этого инструкции цикла либо выполняются еще раз для нового значения счетчика (на основе приведенных выше условий), либо цикл завершается и выполнение продолжается с инструкции, следующей за инструкцией ***Next***.

Изменение значения переменной-счетчика внутри цикла усложняет чтение и отладку программы.

Альтернативный способ выхода из цикла предоставляет инструкция ***Exit For***. Эта инструкция передает управление инструкции, непосредственно следующей за инструкцией ***Next***. Данная команда обычно используется в сочетании с оператором ветвления, в котором проверяется дополнительное условие выхода из цикла.

Допускается организация вложенных циклов ***For...Next*** (один цикл ***For...Next*** располагается внутри другого). Счетчик каждого цикла должен иметь уникальное имя.

Инструкция ***For Each...Next*** повторяет набор инструкций для всех объектов семейства или для всех элементов массива. Visual Basic автоматически определяет значения переменной, выбирая их последовательно из семейства или массива при каждом повторении цикла. Цикл повторяется, пока есть возможность выбора очередного значения для переменной-параметра из группы. Такой цикл записывается по следующему правилу:

```
For Each ПараметрЦикла In Группа
    Код
Next СчетчикЦикла
```

где группа – семейство объектов или массив.

Например, в следующем операторе закрываются все формы, за исключением формы, содержащей текущую процедуру:

```
For Each frm In Application.Forms
    If frm.Caption <> Screen.ActiveForm.Caption Then
       frm.Close
Next
```

В следующем примере выполняется цикл для всех элементов описанного в программе массива, их значения присваиваются индексной переменной I и суммируются в переменной **S**:

```
For Each I In TestArray
   S = S + I
Next I
```

Ниже приведен пример цикла по диапазону ячеек A1:D10 на текущем рабочем листе. Любому числу, записанному в ячейке, имеющему абсолютное значение меньше 0.01, присваивается значение 0 (ноль):

```
For Each myObject In Range("A1:D10")
  If Abs(myObject.Value) < 0.01 Then myObject.Value = 0
Next
```

Допускается выход из цикла ***For Each...Next*** с помощью инструкции ***Exit For***.

Оператор цикла ***For*** дает возможность повторений действий заданное количество раз или для заданного числа элементов, но в VBA существуют более мощные структуры повторения, позволяющие повторять цикл в зависимости от выполнения определенных разработчиком условий. Одним из таких операторов является оператор цикла «с предусловием» (условие, определяющее необходимость повторения цикла, записывается и проверяется до кода действий, подлежащих повторению) оператор ***While ... WEnd***.

Правило записи этого оператора имеет вид:

```
While Условие
     Код
Wend
```

Этот оператор выполняет последовательность инструкций (код), пока заданное условие имеет значение ***True***. Если условие имеет значение ***True***, выполняются все инструкции до инструкции ***Wend***. Затем управление возвращается инструкции ***While*** и вновь проверяется условие. Если условие по-прежнему имеет значение ***True***, процесс повторяется. Если условие получает значение ***False***, выполнение возобновляется с инструкции, следующей за инструкцией ***Wend***. Таким образом, чтобы цикл завершил работу, необходимо включить в записанный в нем код операторы, влияющие на выполнение условия, управляющего циклом.

Циклы ***While...Wend*** могут иметь любую глубину вложенности. Каждая инструкция ***Wend*** соответствует предшествующей инструкции ***While***.

Еще более мощные возможности по организации циклов обеспечивает инструкция ***Do...Loop***, имеющая несколько вариантов. Этот оператор дает возможность организации цикла как «с предусловием», так и «с постусловием», действия могут повторяться как при выполнении условия, так и при его невыполнении.

Правило записи цикла ***Do...Loop*** «с предусловием» имеет вид:

```
Do { While | Until } Условие
   Код
Loop
```


Этот цикл повторяет выполнение набора инструкций (кода), пока условие имеет значение ***True*** (если в заголовке цикла указано ключевое слово ***While*** – «пока») или пока оно не примет значение ***True*** (если в заголовке цикла записано ключевое слово ***Until*** – «пока не»).

Код, записанный в цикле «с предусловием», может выполняться бесконечное число раз или не выполняться ни разу. Если же использовать цикл «с постусловием», в котором сначала записываются операторы, подлежащие повторению в цикле, а потом управляющее повторениями условие, то содержащийся в цикле код должен будет выполняться хотя бы один раз. Для организации цикла «с постусловием» используется следующее правило:

```
Do
    Код
Loop { While | Until } Условие
```

В любом месте управляющей структуры ***Do...Loop*** может быть размещено любое число инструкций ***Exit Do***, обеспечивающих альтернативные возможности выхода из цикла ***Do...Loop***. Часто используемая вместе с определением некоторого условия (например, ***If...Then***), инструкция ***Exit Do*** передает управление инструкции, непосредственно следующей за инструкцией ***Loop***.

Во вложенных циклах ***Do...Loop*** инструкция ***Exit Do*** передает управление циклу охватывающего уровня по отношению к циклу, в котором она вызывается.

При организации цикла с проверкой условия после выполнения кода этот код проработает по крайней мере один раз. Это удобно для организации ввода информации, если требуется проверить правильность ввода:

```
Do
   S = InputBox (prompt := "Введите возраст ребенка:" _
         & Chr(13) & Chr(10) & "(до 16 лет)", _
   Title := "Определение возраста")
Loop Until (Val(S) > 0) And (Val(S)  <= 16)
```

В приведенном примере ответ будет принят, если пользователь введет значение, удовлетворяющее заданному условию (возраст ребенка не может быть отрицательным, да и человек

старше 16 – уже не ребенок). Если же пользователь введет «неправильное» значение, ввод будет повторен.

Еще один оператор очень часто используется в программах на VBA – инструкция ***With***. Этот оператор позволяет использовать более короткие ссылки на объекты, сокращает объем вводимого кода. Более подробную информацию о нем можно получить в справочной системе.

6.3. Запуск программ

Как уже было сказано, для создания процедуры можно записать макрос средствами приложения Microsoft Office или создать процедуру с помощью Редактора Visual Basic, записав ее в модуль, включенный в состав проекта. Записанные процедуры можно выполнять. Для запуска процедур на выполнение существует несколько способов:

– Из окна VBE:
  - поместить точку вставки внутрь тела процедуры и выполнить команду **Запуск программы / UserForm** в меню **Запуск**;
  - поместить точку вставки внутрь тела процедуры и щелкнуть кнопку ▶ (**Запуск программы / UserForm**) на панель инструментов «Стандарт» (эта панель открывается через команду **Панели инструментов** меню **Вид**);
  - поместить точку вставки внутрь тела процедуры и нажать клавишу *F5*;
  - с помощью команды **Макросы…** меню **Сервис** открыть диалоговое окно, из открывшегося списка выбрать нужный макрос и щелкнуть кнопку **Выполнить**.

– Из окна приложения:
  - с помощью команды **Макрос ▸ Макросы** меню **Сервис** открыть диалоговое окно, в открывшемся списке выбрать нужный макрос и щелкнуть кнопку **Выполнить**.

– При запуске приложения Microsoft Office:
  - добавить нужный макрос в обычный шаблон и, щелкнув правой кнопкой мыши по ярлыку, созданному для

записи приложения на рабочем столе, или кнопке панели Microsoft Office, выбрать строку **Свойства;** в поле «Файл» на вкладке «Ярлык» открывшегося диалогового окна добавить к символьной строке, указывающей файл приложения Microsoft Office, строку ***/m* ИмяМакроса**.

Последний способ позволяет автоматически запускать этот макрос при запуске приложения. Например, строка

«C:\Program Files\Microsoft Office\Office\WINWORD.EXE» /m OpenFile

введенная в поле «Файл» диалогового окна свойств приложения, инициирует запуск макроса OpenFile при запуске приложения Word. Текст макроса может быть следующим:

```
Sub OpenFile()
   Dialogs(wdDialogFileOpen).Show
End Sub
```

(этот макрос открывает диалоговое окно «Открытие документа»).

Параметр ***/m*** можно использовать для предотвращения выполнения макросов при запуске приложения (это может быть полезно, если обычный шаблон заражен вирусами).

Открыть конкретный документ (файл) приложения при его запуске можно с помощью параметра ***/f***, указав за ним имя нужного файла.

В приложении макрос (подпрограмма) может быть назначен и кнопке на панели инструментов, и комбинации клавиш. Функцию можно вызывать при организации вычислений в приложении (например, использовать в формуле, записанной в ячейке таблицы Excel, в выражениях, вычисляемых в подпрограммах).

Если программу нужно остановить (например, программа зациклилась в результате ошибки, допущенной при организации цикла), можно также использовать несколько способов:

– нажать клавишу *Esc* (возможно, потребуется многократное нажатие, так как при выполнении программы могут вызываться диалоговые окна, которые тоже обрабатывают нажатие этой клавиши);

– нажать комбинацию клавиш *Ctrl+Break* (установка свойства ***EnableCancelKey*** приложения управляет использованием этой комбинации клавиш);
– если подпрограмма запущена в окне VBE, можно воспользоваться командой **Прервать** меню **Запуск**.

В открывшемся диалоговом окне пользователь может выбрать подходящий вариант действий (продолжить выполнение программы, завершить выполнение программы, перейти в режим отладки или получить справку). Щелчок по кнопке **Отладка** переводит программу в режим прерывания, который рассматривается ниже.

Если при выполнении программы на экран выводится сообщение об ошибке, информацию о ней можно получить в справочной системе. Для поиска ошибки можно осуществить трассировку программы (пошаговое выполнение).

6.4. Отладка программ

В редактор VBE включены средства отладки, которые позволяют осуществить *пошаговое исполнение программы* (выполнение программы приостанавливается после исполнения каждой инструкции), добавить в текст программы *точки останова* (до достижения этой точки программа выполняется в обычном режиме, а при обнаружении точки останова переходит в режим прерывания), отслеживать *контрольные значения* при пошаговом выполнении программы.

Для работы с перечисленными средствами в редакторе VBA имеется меню и панель «Отладка».

Более подробно эти средства рассматриваются в разделе «Инструкции по интерфейсу Visual Basic».

Если не удается запустить на выполнение подпрограмму или макрос в приложении, следует проверить, завершено ли выполнение запущенных ранее программ.

6.5. Обработка ошибок

Обработка ошибок – это определение реакции на выполнение ошибок, которые возникают при выполнении программы. Причины ошибок могут быть как в самой программе, так и вне

ее (например, может отсутствовать нужный файл). Все ошибки можно отнести к трем категориям:

– ошибки *компиляции*, возникающие, когда VBA не может интерпретировать введенный текст;
– ошибки *выполнения*, возникающие после успешной компиляции уже при выполнении программы (причиной этих ошибок могут быть, например, неправильные данные, введенные пользователем);
– *логические* ошибки, которые не распознаются VBA и не приводят к прекращению выполнения программы, но ведут к получению неверных результатов; логические ошибки можно выявить с помощью средств отладки (пошагового выполнения программы, отслеживания контролируемых значений и т.п.).

В VBA есть несколько средств обработки ошибок:
– оператор ***On Error***,
– оператор ***Resume***,
– оператор ***Error***,
– функция ***IsError***,
– функция ***CVErr***,
– объект ***Err***.

Оператор ***On Error*** предназначен для обработки ошибок. Он задает действия, которые должны быть выполнены при возникновении ошибки. Если в программе не выполнена инструкция ***On Error***, то любая ошибка выполнения является фатальной; это означает, что выводится сообщение об ошибке и выполнение программы прекращается. Этот оператор заставляет программу реагировать на ошибку различным образом.

Подпрограмма обработки ошибок не может быть процедурой ***Sub*** или ***Function***. Эта подпрограмма должна быть частью программы, которая отмечается с помощью метки строки или номера строки.

Оператор

On Error GoTo **Метка**

можно использовать как общий обработчик ошибок (в случае возникновения ошибки управление будет передано на оператор, которому предшествует указанная метка, следующие за ней ин-

струкции могут проверить тип возникшей ошибки, уточнить причину и выполнить действия по ее устранению).

«Включенным» обработчиком ошибок называют подпрограмму, которая указана в инструкции ***On Error***; «активным» обработчиком ошибок является включенный обработчик ошибок, который обрабатывает текущую ошибку. Если ошибка возникает в самом обработчике ошибок (в промежутке между возникновением ошибки и выполнением инструкции ***Resume***, ***Exit Sub***, ***Exit Function*** или ***Exit Property***), то обработчик ошибок, определенный в текущей процедуре, не может обработать ошибку. Управление в этом случае возвращается в вызывающую процедуру; если в вызывающей процедуре включен обработчик ошибок, то обработка ошибки передается ему. Если этот обработчик ошибок является в данный момент активным, т.е. уже обрабатывает ошибку, то управление снова передается назад в вызывающую процедуру и т.д. до тех пор, пока не будет найден включенный, но не активный обработчик ошибок. Если включенный, но неактивный обработчик ошибок найден не будет, ошибка становится фатальной в том месте программы, в котором она впервые возникла. При каждой передаче управления обработчиком ошибок в вызывающую процедуру эта процедура становится текущей. После завершения обработки ошибки обработчиком в любой процедуре возобновляется выполнение текущей процедуры с той ее части, которая указана в инструкции ***Resume***.

Для определения причины ошибки в подпрограммах обработки ошибок используют значение свойства ***Number*** объекта ***Err***. Необходимо обеспечить в подпрограммах обработки ошибок проверку или сохранение существенных значений свойств объекта ***Err*** перед тем, как может возникнуть новая ошибка или перед вызовом процедуры, в которой может возникнуть новая ошибка. Значения свойств объекта ***Err*** описывают последнюю ошибку. Текст сообщения об ошибке, соответствующего коду ошибки ***Err.Number***, содержится в свойстве ***Err.Description***.

Для отключения общей обработки следует использовать оператор ***On Error GoTo*** **0**. Конструкция ***On Error GoTo*** **0** отключает обработку ошибок в текущей процедуре. Эта конструкция не задает переход на строку **0** для обработки ошибок, даже если

в процедуре имеется строка с номером **0**. Если инструкция ***On Error GoTo*** **0** не выполнялась, то обработчик автоматически отключается при выходе из процедуры.

Инструкция

On Error Resume Next

указывает, что возникновение ошибки выполнения приводит к передаче управления на инструкцию, непосредственно следующую за инструкцией, при выполнении которой возникла ошибка, или на инструкцию, непосредственно следующую за вызывающей инструкцией в процедуре, содержащей конструкцию ***On Error Resume Next***, т.е. эта инструкция приводит к игнорированию ошибки. Это позволяет продолжить исполнение программы, несмотря на ошибку выполнения. Это позволяет также встроить подпрограмму обработки ошибок в процедуру, а не передавать управление в другую часть процедуры. Конструкция ***On Error Resume Next*** становится неактивной при вызове новой процедуры, поэтому для внутренней обработки ошибок необходимо выполнять инструкцию ***On Error Resume Next*** в каждой вызываемой процедуре. Этой формой оператора следует пользоваться с осторожностью, так как ошибка фактически не обрабатывается. Рекомендуется при доступе к объектам использовать эту форму инструкции, а не ***On Error GoTo***.

При обработке ошибок, возникающих при доступе к другим объектам, рекомендуется использовать конструкцию ***On Error Resume Next***. Проверка объекта ***Err*** после каждого взаимодействия с другим объектом позволяет устранить неопределенность в том, при доступе к какому объекту возникла ошибка. Это позволяет всегда точно знать, какой объект поместил значение кода ошибки в свойство ***Err.Number***, а также в каком объекте возникла ошибка (эта информация содержится в свойстве ***Err.Source***).

Для того чтобы предотвратить выполнение программы обработки ошибок в тех случаях, когда ошибка не возникла, следует помещать соответствующую инструкцию (***Exit Sub***, ***Exit Function*** или ***Exit Property***) перед подпрограммой обработки ошибки, как в следующем примере:

```
Sub InitializeMatrix(Var1, Var2, Var3, Var4)
   On Error GoTo ОбработкаОшибок
```

```
    ‘   Код процедуры:
    . . .
    Exit Sub
    ОбработкаОшибок:
    . . .
    Resume Next
End Sub
```

В этом примере программа обработки ошибок помещена между инструкциями ***Exit Sub*** и ***End Sub***, что позволяет отделить ее от части программы, соответствующей нормальному выполнению процедуры. Программу обработки ошибок можно разместить в любом месте процедуры.

Системные ошибки при вызовах библиотек динамической компоновки (DLL) не приводят к возникновению исключений и не перехватываются средствами Visual Basic. При вызове функций из библиотек DLL необходимо проверять, успешно ли возвращается каждое значение (согласно спецификациям API), и в случае неудачи проверять значение свойства ***LastDLLError*** объекта ***Err***.

Оператор ***Resume*** используется для продолжения выполнения кода с указанной строки. Эта инструкция может использоваться только в процедурах обработки ошибок. Она имеет несколько форматов (***Resume***, ***Resume Next***, ***Resume*** **Строка**).

Инструкция ***Error*** имитирует возникновение ошибки с кодом, переданным ей в качестве аргумента, заполняет свойства объекта ***Err*** значениями по умолчанию, после чего вызывает обработчик ошибок. Эта инструкция поддерживается для совместимости с предыдущими версиями. В новой программе, в особенности при создании объектов, следует применять для создания ошибок выполнения метод ***Raise*** объекта ***Err***.

Функция ***CVErr*** используется для преобразования числового значения, переданного функции в качестве аргумента, в код ошибки. Возвращает значение типа ***Variant*** с подтипом ***Error***, содержащее код ошибки, указанный пользователем.

Функция ***CVErr*** применяется для создания ошибок, определяемых пользователем, в создаваемых пользователем процедурах. Например, при создании функции, которая принимает несколько аргументов и обычно возвращает строку, имеется

возможность проверить, что введенные аргументы попадают в допустимый диапазон значений. Если это не так, весьма вероятно, что функция не вернет ожидаемое значение. В этом случае функция ***CVErr*** позволяет вернуть код ошибки, чтобы знать, какие действия необходимо предпринять.

При использовании этой функции неявное преобразование значения Error не допускается. Например, не допускается прямое присвоение возвращаемого значения ***CVErr*** переменной с типом, отличным от ***Variant***. Однако имеется возможность выполнить явное преобразование (с помощью ***CInt***, ***CDbl*** и т.п.) значения, возвращаемого ***CVErr***, и присвоить это значение переменной с соответствующим типом данных.

В приведенном ниже примере функция **WhichRange** проверяет введенное пользователем строковое значение переменной **UserText** на возможность его преобразования в число, принадлежащее диапазону от 1 до 10. Если преобразование не может быть выполнено или полученное число не попадает в заданный диапазон, в качестве результата вырабатывается код ошибки.

```
Sub CVErr_Example()
  Dim VarResult As Variant
  Dim UserText As String
‘ Код для ввода значения переменной UserText:
   . . .
‘ Вызов функции для преобразования значения переменной:
 VarResult = WhichRange(UserText)
‘ Проверка, является ли результат кодом ошибки:
 If IsError(VarResult) Then
   MsgBox  Prompt := “Нужно ввести число от 1 до 10”, _
           Title := “Ошибка ввода”
 Else
   MsgBox  Prompt := “Введено число ” & Str(VarResult)
 End If
End Sub

‘ Описание функции:
Function WhichRange(ByVal Number As Integer) As Variant
 If Number >= 1 And Number <=10 Then
   WhichRange = Number
 Else
   WhichRange = CVErr(200)
 End If
End Function
```

Объект ***Err*** предназначен для управления информацией, связанной с ошибками. Свойства этого объекта задаются генератором ошибок Visual Basic, объектом или разработчиком программы Visual Basic. Свойством объекта, используемым по умолчанию, является свойство ***Number***.

Если возникает ошибка выполнения, то свойства объекта ***Err*** получают значения, однозначно определяющие ошибку. В эти свойства записываются также сведения, которые могут быть использованы для обработки ошибки. Для того чтобы создать ошибку выполнения в программе, следует использовать метод ***Raise***.

Значения свойств объекта ***Err*** сбрасываются до нулей или пустых строк (*""*) после выполнения инструкций ***Resume*** или ***On Error*** любого вида, а также после выполнения инструкций ***Exit Sub***, ***Exit Function*** или ***Exit Property*** в программе обработки ошибки. Для того чтобы явным образом сбросить значения свойств объекта ***Err***, следует вызвать метод ***Clear***.

Более подробную информацию об обработке ошибок и тонкостях использования различных форм описанных операторов можно получить в справочной системе.

6.6. Преобразования типов

VBA содержит процедуры, позволяющие преобразовывать значение одного типа в значение другого типа. Функции преобразования можно поделить на три категории: общие преобразования строковых и числовых значений, преобразования строковых и числовых значений в значения конкретных типов, преобразования числовых значений в значения ошибки.

Общие преобразования строковых и числовых значений осуществляются с помощью следующих функций:

- ***Val*** – преобразование строки в число;
- ***Str*** – преобразование числа в строку;
- ***Hex*** – преобразование числа в шестнадцатеричное число;
- ***Oct*** – преобразование числа в восьмеричное число.

Группу функций преобразования в значения конкретных типов (функций *приведения типов*) составляют следующие функции: ***CBool***, ***CByte***, ***CCur***, ***CDate*** (***CVDate***), ***CDbl***, ***CInt***, ***Int***, ***Fix***, ***CLng***, ***CSng***, ***CStr***, ***Str***, ***CVar***. Эти функции гарантируют, что полученные с их помощью значения будут принадлежать конкретным типам данных. Они корректно учитывают национальные установки для записи числовых значений (кроме ***Int***, ***Fix***, ***Str*** – они используют в качестве разделителя только точку, т.е. не учитывают национальные установки) и значений даты/времени (эти параметры устанавливаются с помощью опции «Язык и стандарты» на Панели управления Windows).

Преобразование чисел в коды ошибок рассматривалось выше.

Более детальную информацию об использовании перечисленных функций можно получить в справочной системе VBA.

6.7. Функции для работы с датами и временем

VBA поддерживает несколько функций для работы с датами и временем. Эти функции используют внутренние часы компьютера для получения текущей даты или времени (их настройка осуществляется с помощью опции «Дата/Время» Панели управления Windows). Ниже перечислены основные функции:

- ***Date*** – возвращает (при использовании в качестве функции) или устанавливает (при вызове в качестве подпрограммы, т.е. при использовании как оператора) текущую дату;
- ***Now*** – возвращает текущую дату и время, установленные в ПК;
- ***Time*** – возвращает (при вызове в качестве функции) или устанавливает (при использовании в качестве отдельной инструкции вызова подпрограммы) текущее время;
- ***Timer*** – возвращает количество секунд, прошедших после полуночи (используется для хронометризации выполнения кода), функция ***GetTickCount*** из Win32 API дает более точный результат;
- ***DateAdd*** – возвращает дату/время, отстоящую от текущего времени на заданный интервал, который может быть задан в годах, кварталах, месяцах, неделях, днях, часах, минутах, секундах;

– ***DateDiff*** – возвращает величину интервала времени между двумя датами;
– ***DatePart*** – возвращает указанный компонент даты/времени (т.е. секунды, минуты, часы и т.д.);
– ***DateSerial*** – преобразует три числа: год, месяц и день в значение даты/времени;
– ***DateValue*** – возвращает значение даты/времени для строки, представляющей дату;
– ***Day*** – возвращает день месяца для заданного значения даты/времени;
– ***Hour*** – возвращает часы для заданного значения даты/времени;
– ***Minute*** – возвращает минуты для заданного значения даты/времени;
– ***Month*** – возвращает месяц для заданного значения даты/времени;
– ***Second*** – возвращает секунды для заданного значения даты/времени;
– ***TimeSerial*** – преобразует три числа: часы, минуты и секунды в значение даты/времени;
– **TimeValue** – возвращает значение даты/времени для строки, представляющей время;
– ***WeekDay*** – возвращает день недели (от 1 до 7);
– ***Year*** – возвращает год для заданного значения даты/времени.

Функции для работы с датами/временем возвращают значения в формате, который установлен на Панели управления Windows (опция «Язык и стандарты»). Для установки пользовательского формата можно использовать функцию ***Format***.

Более подробную информацию о перечисленных функциях можно найти в разделе «Функции» справочного руководства по Visual Basic. Для каждой функции дополнительную полезную информацию можно получить, щелкнув мышкой по строке «См. также» (например, для функции ***Format*** можно просмотреть список специальных форматов).

6.8. Управление файлами

В VBA есть набор процедур для работы с файлами. Основные средства управления файлами перечислены ниже:

– ***ChDir*** – оператор изменения текущей папки (каталога);
– ***ChDrive*** – оператор изменения текущего диска;
– ***Close*** – оператор закрытия файла, открытого с помощью оператора ***Open***;
– ***CurDir*** – функция, возвращающая в качестве результата текущую папку;
– ***Dir*** – функция, использующаяся для проверки существования файла;
– ***EOF*** – функция проверки, достигнут ли конец файла, открытого с помощью ***Open***;
– ***FileAttr*** – функция определения статуса файла, открытого с помощью оператора ***Open***;
– ***FileCopy*** – оператор копирования файлов;
– ***FileDateTime*** – функция, возвращающая дату и время создания или последней модификации файла;
– ***FileLen*** – функция, возвращающая размер (в байтах) закрытого файла;
– ***FreeFile*** – функция определения доступного номера файла, который может использовать оператор ***Open***;
– ***GetAttr*** – функция, возвращающая атрибуты файла;
– ***SetAttr*** – оператор, устанавливающий атрибуты файла;
– ***Kill*** – оператор удаления файлов;
– ***Loc*** – функция определения текущей позиции чтения/записи в файле, открытом с помощью оператора ***Open***;
– ***Seek*** – оператор установки текущей позиции чтения/записи в файле, открытом с помощью оператора ***Open***;
– ***LOF*** – функция, возвращающая размер (в байтах) открытого с помощью оператора ***Open*** файла (для текстовых файлов – число символов);
– ***MkDir*** – оператор создания папки (каталога);
– ***RmDir*** – оператор удаления папки (каталога);

– ***Open*** – оператор открытия существующего файла или создания нового файла; в операторе указывается режим работы с файлом: ***Append*** (8), ***Binary*** (32), ***Input*** (1), ***Output*** (2), ***Random*** (4);
– ***Reset*** – оператор сохранения и закрытия всех файлов, открытых с помощью оператора ***Open***.

Более подробную информацию об использовании этих операций с файлами можно получить в справочной системе.

6.9. Функции проверки

Функции этой группы позволяют получать информацию о данных, используемых в приложении:

– ***IsArray*** – возвращает значение типа ***Boolean***, показывающее, является ли переменная-аргумент массивом;
– ***IsDate*** – возвращает значение типа ***Boolean***, показывающее, может ли значение выражения-аргумента быть преобразовано в значение даты;
– ***IsEmpty*** – возвращает значение типа ***Boolean***, показывающее, была ли инициализирована переменная-аргумент функции;
– ***IsError*** – возвращает значение типа ***Boolean***, показывающее, представляет ли выражение-аргумент значение ошибки;
– ***IsMissing*** – возвращает значение типа ***Boolean***, показывающее, был ли передан в процедуру необязательный аргумент;
– ***IsNull*** – возвращает значение типа ***Boolean***, показывающее, является ли результатом вычисления выражения-аргумента пустое значение (***Null***);
– ***IsNumeric*** – возвращает значение типа ***Boolean***, показывающее, имеет ли выражение (аргумент функции) числовое значение;
– ***IsObject*** – возвращает значение типа ***Boolean***, показывающее, представляет ли идентификатор объектную переменную;
– ***Err*** – возвращает номер ошибки;

– ***IMEStatus*** – возвращает текущее значение Input Method Editor;
– ***QBColor***, ***RGB*** – возвращают информацию о цвете;
– ***TypeName***, ***TypeVar*** – возвращают название типа и подтипа указанной переменной.

Более подробную информацию можно получить в разделе «Функции» справочного руководства по Visual Basic.

6.10. Функции работы со строками

VBA имеет множество функций для работы со строками. Основные функции приведены ниже:

– ***Asc***, ***AscB***, ***AscW*** – функции, возвращающие код первого символа в строке;
– ***Chr***, ***ChrB***, ***ChrW*** – функции, возвращающие символ с заданным кодом;
– ***Format*** – используется для форматирования данных;
– ***InStr***, ***InStrB*** – используются для поиска вхождений одной строки в другую;
– ***LCase*** – используется для преобразования всех символов в строке к нижнему регистру (строчным буквам);
– ***Left***, ***LeftB*** – возвращает указанное количество символов от начала строки;
– ***Len***, ***LenB*** – возвращает длину строки;
– ***LTrim***, ***RTrim***, ***Trim*** – функции для удаления лишних пробелов;
– ***Mid***, ***MidB*** – возвращают указанное количество символов, начиная с заданной позиции внутри строки;
– ***Right***, ***RightB*** – возвращают указанное количество символов от конца строки.

Более детальную информацию об использовании этих функций можно получить в разделе «Функции» справочного руководства по Visual Basic.

6.11. Взаимодействие с пользователем

Существуют достаточно простые средства для организации взаимодействия пользователя с приложением. Основные элементы интерфейса приведены ниже:

- ***AppActivate*** – оператор активизации открытого приложения (его окна);
- ***Beep*** – подача звукового сигнала;
- ***Choose*** – выбор элемента из списка;
- ***CreateObject*** – создает и возвращает ссылку на объект ActiveX;
- ***GetObject*** – возвращает ссылку на объект ActiveX, сохраненный в файле;
- ***DeleteSetting***, ***GetAllSettings***, ***GetSetting***, ***SaveSetting*** – используются для установки и обновления информации о приложении в реестре;
- ***DoEvents*** – приостанавливает выполнение приложения, чтобы система смогла обработать другие события;
- ***IIf*** – функция, являющаяся упрощенной версией структуры управления ***If..Then..Else***;
- ***InputBox*** – используется для получения информации от пользователя;
- ***MsgBox*** – используется для вывода информации;
- ***SendKeys*** – используется для моделирования нажатия пользователем последовательности клавиш;
- ***Shell*** – используется для запуска других приложений;
- ***Switch*** – функция, расширяющая возможности функции ***IIf*** (позволяет рассматривать не две, а большее количество альтернатив).

Некоторые из приведенных функций подробнее рассматриваются ниже. Более подробную информацию об использовании других функций можно получить в справочной системе.

6.12. Пользовательские формы и диалоговые окна

Приложения используют диалоговые окна, когда от пользователя требуется ввести информацию для выполнения приложением различных задач. Диалоговое окно можно создать на основе пользовательской формы.

Средства VBA позволяют создавать формы. Для создания формы в меню **Вставка** Редактора Visual Basic нужно выбрать команду **UserForm**. После ее выполнения появится пустая форма и панель инструментов «Панель элементов» с набором элементов управления, которые могут быть размещены на форме (рис. 6.2).

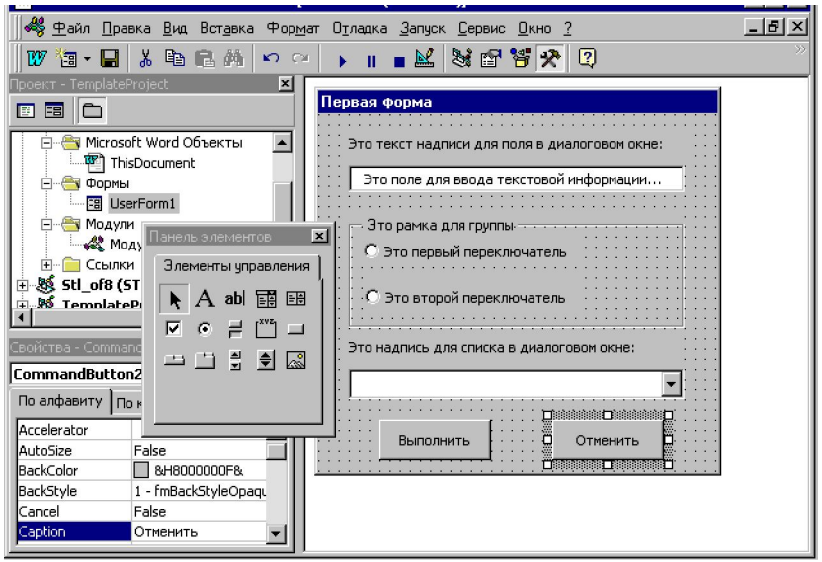


Рис. 6.2. **Средства разработки форм**

Элементы управления используются для придания форме функциональных возможностей. Элементы управления могут быть различными по сложности, начиная с простых командных кнопок и до элементов управления мультимедиа.

Стандартными элементами управления VBA являются следующие элементы: Надпись (***Label***), Поле (***TextBox***), Поле со списком (***ComboBox***), Список (***ListBox***), Флажок (***CheckBox***), Переключатель (***OptionButton***), Выключатель (***ToggleButton***), Рамка (***Frame***), Кнопка (***CommandButton***), Набор вкладок (***TabStrip***), Набор страниц (***MultiPage***), Полоса прокрутки (***ScrollBar***), Счетчик (***SpinButton***), Рисунок (***Image***).

Для добавления элемента в пользовательскую форму его нужно выделить на панели элементов и с помощью мыши разместить в нужном месте формы, «вытянув» до необходимого размера.

Для установки свойств элементов управления вручную необходимо выделить элемент управления и открыть окно свойств через меню **Вид** (если оно закрыто).

Пользовательские формы позволяют построить *управляемый событиями* интерфейс приложения. В таком приложении пользователь управляет его работой, нажимая кнопки или клавиши, а программа реагирует на возникающие при этом события.

Просмотреть список событий для выделенного элемента формы можно, выбрав команду **Программа** в меню **Вид** и раскрыв список процедур справа вверху (слева показано имя выделенного элемента). После этого появится список событий для выделенного элемента.

После выбора события из списка Редактор VBA автоматически создаст заготовку процедуры обработки этого события. Двойной щелчок по нужному элементу на форме приводит к созданию заготовки процедуры обработки события по умолчанию (для командной кнопки, например, это событие ***Click***).

Процедуры обработки событий являются скрытыми (***Private***), т.е. могут запускаться только из пользовательской формы, которая их содержит, они используются только для определения реакции на события для данного элемента управления, поэтому им назначаются имена вида *ИмяЭлементаУправления_Событие*. Процедура обработки события запускается автоматически, как только происходит соответствующее событие.

Хотя все объекты элементов управления имеют свой собственный уникальный набор свойств, методов и событий, существует и *общий набор свойств и событий, применимый ко всем элементам*:

– *общие свойства*:

- ***Name*** – имя элемента управления, по которому на него производится ссылка в коде;

- ***AutoSize*** – если это свойство установлено в ***True***, элемент управления будет иметь наименьший возможный размер, необходимый для отображения его содержимого;
- ***Enabled*** – если свойство установлено в ***True***, пользователь может взаимодействовать с этим элементом управления, если же значение свойства равно ***False***, элемент управления защищен и на него не может быть установлен «фокус» при работе с формой;
- ***Font*** – название шрифта, его размер и стиль, используемый для отображения текста, связанного с элементом управления (не применяется для полос прокрутки и счетчиков);
- ***Left***, ***Top***, ***Width***, ***Height*** – свойства, определяющие позицию и размер элемента управления в пунктах;
- ***Locked*** – если свойство установлено в ***True***, и, кроме того, включена защита рабочего листа для объектов, то пользователь не может редактировать элемент управления во время разработки;
- ***Visible*** – если установлено в ***True***, элемент отображается во время выполнения, если значение свойства равно ***False***, объект скрыт;

– *общие события*:

- ***Click*** – происходит, когда пользователь нажимает или отпускает кнопку мыши на элементе управления;
- ***DblClick*** – происходит, когда пользователь дважды щелкает кнопкой мыши на элементе управления;
- ***KeyPress*** – происходит при нажатии клавиши ANSI (любой клавиши на клавиатуре, за исключением клавиш управления курсором, функциональных клавиш и клавиш управления *Ctrl, Shift, Alt*);
- ***GotFocus***, ***LostFocus*** – происходит, когда элемент управления получает или теряет «фокус» соответственно;
- ***MouseDown***, ***MouseMove***, ***MouseUp*** – используются для определения отдельных действий при нажатии кнопки мыши (определение точного положения курсора на

экране, где была нажата кнопка мыши, была ли нажата клавиша *Ctrl, Shift* или *Alt*, когда произошло событие);

- ***BeforeDragOver***, ***BeforeDropOrPaste*** – используются для реализации функциональных возможностей перетаскивания в элементе управления;
- ***Error*** – используется для уведомления приложения о том, что произошла ошибка и она относится к элементу управления (предоставляет информацию об ошибке, которую можно обработать в коде).

Более подробную информацию можно получить в справочной системе VBA, выполнив поиск информации об интересующем событии по его названию. Ниже приведена информация об основных свойствах, методах и событиях пользовательских форм (рис. 6.3).

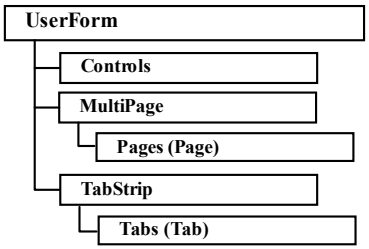


***Рис. 6.3*. Иерархическая схема объекта** *UserForm*

Свойства объекта ***UserForm***:

- ***Name*** или ***CodeName*** – содержит имя, используемое при ссылке на пользовательскую форму в коде;
- ***BackColor*** – цвет фона;
- ***Caption*** – текст, отображаемый в заголовке окна формы;
- ***Left***, ***Top***, ***Height***, ***Width*** – положение и размер окна;
- ***Picture*** – рисунок, отображаемый как фон;
- ***StartUpPosition*** – применяется для автоматического центрирования формы.

Методы объекта ***UserForm***:

- ***Show*** – модально отображает пользовательскую форму;
- ***Hide*** – закрывает пользовательскую форму;
- ***PrintForm*** – печатает изображение формы.

События объекта ***UserForm***:

- ***Initialize*** – происходит при отображении формы;
- ***QueryClose*** – происходит перед закрытием пользовательской формы; с помощью установки аргумента ***Cancel*** в ***True*** можно отменить закрытие формы;
- ***Terminate*** – происходит после закрытия пользовательской формы.

Основные элементы управления в пользовательских формах и *их свойства* (в дополнение к перечисленным выше свойствам):

- ***CommandButton*** – командные кнопки, для которых определены свойства:
  - ***Caption*** – это свойство задает текст, отображаемый на кнопке;
  - ***Picture*** – свойство, определяющее графическое изображение на кнопке;
  - ***TakeFocusOnClick*** – если свойство равно ***False***, кнопка не принимает фокус (он сохраняется при нажатии на кнопку), если ***True*** – фокус переводится на кнопку;
  - ***Default*** – установка этого свойства в ***True*** делает кнопку кнопкой по умолчанию, т.е. при нажатии клавиши *Enter* кнопка активизируется так же, как если бы на ней был произведен щелчок;
  - ***Cancel*** – установка этого свойства в ***True*** делает кнопку отменяющей, т.е. нажатие клавиши *Esc* активизирует кнопку так же, как если бы по ней был выполнен щелчок;
- ***CheckBox*** – флажок – имеет как правило, два состояния (установлен и сброшен), которые соответствуют значениям ***True*** и ***False***, но может быть настроен для трех состояний (установлен, сброшен и не определен); может быть связан с данными приложения (например, с диапазоном ячеек рабочего листа):
  - ***Caption*** – свойство, задающее текст, отображаемый рядом с этим элементом управления;
  - ***TripleState*** – при установке в ***True*** значения этого свойства флажок может принимать три состояния (***True***, ***False***, ***Null***);

- ***Value*** – свойство, определяющее значение флажка;
- ***LinkedCell*** – позволяет автоматически помещать значение флажка в ячейку (это свойство используется при размещении элементов управления на рабочем листе Excel);
- ***GroupName*** – имя группы, в которую включается флажок; группу можно выделить на экране с помощью рамки (объект ***Frame***);

– ***OptionButton*** – переключатель – по свойствам и функциональным возможностям напоминает предыдущий элемент, но элементы этого типа объединяются в группы, в каждой группе может быть установлен лишь один переключатель, по умолчанию все переключатели объединяются в одну группу, а для разделения переключателей по нескольким группам используется свойство ***GroupName***; переключатели, помещенные на пользовательской форме внутри рамки (объекта ***Frame***) будут выполнять роль группы вне зависимости от установки их свойства ***GroupName***; причем при перемещении рамки помещенные в нее переключатели также будут перемещаться;

– ***ToggleButton*** – выключатель – по свойствам аналогичен элементу управления CheckBox, но выглядит на экране, как кнопка, которая может быть нажата или отжата;

– ***ListBox*** – список, позволяющий выделить один или несколько элементов; для него можно изменять размер окна и количество одновременно отображаемых элементов; можно организовать прокрутку элементов с помощью полосы прокрутки; этот управляющий элемент имеет следующие свойства и методы:

- ***List*** – свойство, задающее список, представляет собой массив строк, где каждый элемент – это один элемент списка;
- ***ListCount*** – количество элементов в списке;
- ***MultiSelect*** – определение способа выделения элементов в списке (может принимать одно из следующих значений: ***fmMultiSelectSingle*** – допускается выделение единственного элемента, ***fmMultiSelectMulti*** – простое

многоэлементное выделение (может быть выделено несколько элементов щелчками мыши на каждом из них), ***fmMultiSelectExtended*** – расширенное множественное выделение (элементы выделяются щелчками мыши при нажатых клавишах *Shift* или *Ctrl*));
- ***ListStyle*** – устанавливает стиль изображения списка (***fmListStylePlain*** – список изображается в обычном виде, ***fmListStyleOption*** – отображается столбец флажков или переключателей слева от элементов списка);
- ***ListIndex*** – индекс выделенного в данный момент элемента списка (для списков с возможностью единичного выделения);
- ***Value*** – содержит текст, выделенный в списке в данный момент (используется для списков с единичным выделением);
- ***Selected*** – массив логических значений, каждый элемент которого отражает состояние выделения соответствующего элемента списка (значение элемента этого массива равно ***True***, если соответствующий элемент в списке выделен, и ***False*** – для невыделенного элемента);
- ***ListFillRange*** – используется в Excel для указания диапазона ячеек рабочего листа, с которым связано содержимое ячеек списка;
- ***LinkedCell*** – используется в Excel для указания диапазона ячеек рабочего листа, с которым связано свойство ***Value***, содержащее выделенное в данный момент значение (при единичном выделении);
- ***MatchEntry*** – определяет тип автоматического перемещения по элементам списка при вводе с клавиатуры: ***fmMatchEntryFirstLetter*** – подбирает для каждого введенного символа первую строку в списке, начинающуюся с этого символа, ***fmMatchEntryComplete*** – подбирает строку, содержащую все введенные символы, ***fmMatchEntryNone*** – отключает возможность подбора;
- ***AddItem*** – метод добавления одного элемента в список; в качестве аргументов передаются строка текста,

представляющая элемент (***pvargText***), и индекс элемента в списке (0 – для добавления элемента в вершину);
- ***Clear*** – метод удаления всех элементов из списка;
- ***RemoveItem*** – метод удаления одного или нескольких элементов из списка; имеет один аргумент – индекс первого удаляемого элемента списка (***pvargIndex***, строки нумеруются с 0);

– ***ComboBox*** – поле со списком – элемент, напоминающий по функциональным возможностям рассмотренный выше список, но дает возможность выбирать и вводить только один элемент; имеются следующие особенности:
- ***Style*** – при установке в свойстве ***Style*** значения ***fmStyleDropDownList*** поле со списком становится похожим на список, а установка значения ***fmStyleDropDownCombo*** дополнительно дает возможность пользователю ввести значение в видимое поле, причем этого значения может не быть в списке, тогда введенное значение присваивается свойству ***Value***, но в список не вставляется (его можно будет добавить позднее);
- ***ListRows*** – количество элементов, отображаемых в раскрываемом списке после нажатия кнопки раскрытия списка; все элементы дает возможность просматривать полоса прокрутки;

– ***ScrollBar*** (полоса прокрутки) и ***SpinButton*** (счетчик) – сходные по своим функциональным возможностям элементы управления (счетчик, по сути, является полосой прокрутки без ползунка), предоставляющие графический интерфейс для установки числовых значений; для этих элементов важны следующие свойства, события и методы:
- ***Max*** – наибольшее значение (положительное число или 0);
- ***Min*** – наименьшее значение (положительное число или 0);
- ***SmallChange*** – положительное или отрицательное целое число, задающее величину изменения счетчика или полосы прокрутки при щелчке по одной из стрелок;

- ***LargeChange*** – положительное или отрицательное целое число, задающее величину изменения полосы прокрутки при щелчке внутри полосы прокрутки (для счетчика не используется);
- ***Value*** – текущее значение полосы прокрутки или счетчика;
- ***LinkedCell*** – для элемента на листе Excel можно задать диапазон, с которым связано значение свойства ***Value***;
- ***Change*** – событие, происходящее при изменении значения элемента управления;

– ***TextBox*** – поле, дающее возможность пользователю ввести текст в указанную область; имеет следующие основные свойства:
- ***Text*** – устанавливает или возвращает текст, содержащийся в элементе управления;
- ***MultiLine*** – если установлено в ***True***, элемент управления будет отображать многострочный текст;
- ***MaxLength*** – максимальное количество символов, которые можно ввести в один элемент;
- ***PasswordChar*** – символ, отображаемый при вводе «пароля» (если это свойство определено, данный символ будет отображаться вместо вводимого текста);
- ***LinkedCell*** – позволяет на рабочем листе Excel автоматически помещать значение свойства Text в указанную ячейку;

– ***Frame*** – рамка, используется для визуальной группировки элементов управления на форме; на рабочем листе элементы можно сгруппировать с помощью элемента ***GroupBox***;

– ***Label*** – надпись, используется для отображения текстов на форме; наиболее важное свойство этого элемента управления – ***Caption*** – задает отображаемый текст;

– ***Image*** – рисунок, используется для отображения графики;

– ***MultiPage*** – набор страниц – элемент, доступный только на пользовательских формах, используется для реализации многостраничных диалоговых окон; каждая страница имеет свой ярлычок, на каждую страницу можно поместить свои элементы управления; при разработке формы

можно создать новую страницу, удалить, переместить или переименовать существующую страницу с помощью команд контекстного меню, вызываемого щелчком правой кнопкой мыши по ярлычку страницы; элементы управления размещаются на каждой странице независимо от других страниц; могут быть также элементы управления, не принадлежащие ни одной странице, размещенные непосредственно на форме;

– ***TabStrip*** – набор вкладок – элемент, используемый для создания диалоговых окон с вкладками; отличие от многостраничного окна в том, что этот элемент не может содержать других элементов управления и необходимо писать код, отображающий и скрывающий элементы управления для каждой вкладки, в обработчике события, происходящего при переключении пользователем вкладок, а это сложнее, чем просто поместить элементы на форму и написать обработчики событий для них;
– ***RefEdit*** – элемент, используемый только в пользовательских формах, созданных в Excel, позволяет указывать ячейки в модальном диалоговом окне для выделения диапазона; он имеет кнопку справа, с помощью которой можно временно скрыть диалоговое окно и увидеть содержимое рабочего листа; основное свойство этого элемента – ***Text***.

Последовательность перехода определяет порядок, в котором элементы диалогового окна будут активизироваться при нажатии клавиши *Tab*. Этот порядок должен быть естественным и понятным пользователю. Для изменения последовательности перехода можно в Редакторе VBA выбрать команду **Последовательность перехода** в меню **Вид**. В открывшемся диалоговом окне будет отображена текущая последовательность. Для ее изменения следует выбрать нужный элемент и перемещать его вверх или вниз в списке с помощью соответствующих кнопок.

Для отображения пользовательской формы из процедуры VBA вызывается метод ***Show*** пользовательской формы. После закрытия формы можно проверить «причину» ее закрытия. В приведенном ниже примере пользовательская форма **UserForm1** может быть закрыта щелчком по кнопке **Выполнить**

(первая командная кнопка – **CommandButton1**) или кнопке **Отменить** (вторая командная кнопка – **CommandButton2**). После ее закрытия в процедуре проверяется, какая кнопка была нажата:

```
Sub UserForm_Example()
   UserForm1.Show
   If  UserForm1.Tag = vbOK Then
      MsgBox "Нажата кнопка «Выполнить»"
   Else
      MsgBox "Нажата кнопка «Отменить»"
   End If
End Sub
```

Чтобы закрыть пользовательскую форму, можно вызвать метод ***Hide*** элемента управления на пользовательской форме. Пользовательская форма имеет модальный характер, поэтому в одной подпрограмме нельзя и открыть, и закрыть форму. В модуле кода формы находятся обработчики событий для ее элементов. Ниже приведен код обработчиков события ***Click*** для двух командных кнопок. Кроме закрытия формы в этих обработчиках устанавливается свойство ***Tag***. Это делается для того, чтобы после закрытия диалогового окна можно было определить, какая кнопка была нажата.

```
Private Sub CommandButton1_Click()
     Me.Hide
     Me.Tag = vbOK
End Sub

Private Sub CommandButton2_Click()
     Me.Hide
     Me.Tag = vbCancel
End Sub
```

Ключевое слово ***Me*** является ссылкой на объект, содержащий выполняемый код (в данном случае – на пользовательскую форму **UserForm1**). Свойство ***Tag*** используется для сохранения какой-либо полезной информации о форме (например, о событиях, предшествующих закрытию формы).

Параметры элементов управления, размещенных на форме, можно задать при ее инициализации (при обработке события ***Initialize***, происходящего при отображении формы) или непосредственно перед вызовом метода ***Show*** формы. Например, сле-

дующий код создает пользовательскую форму (рис. 6.4) для замены значений в строке с номером 5 рабочего листа Excel:

```
With UserForm1
   .StartUpPosition = CenterScreen
   .TextBox1.Text = 0
   .TextBox2.Text = 1000
   .CheckBox1.Caption = _
             "Выводить запрос перед заменой значения"
   .CheckBox1.TripleState = False
   .CheckBox1.Value = False
   .CommandButton1.Default = True
   .CommandButton2.Cancel = True
   .Show
   F = Val(.TextBox1.Text)
   T = Val(.TextBox2.Text)
   S = .CheckBox1.Value
End With
If UserForm1.Tag = vbOK Then
   For Each i In [A5:J5]
      i.Select
      If i.Value = F Then
         If S Then
            x = MsgBox("Выполнить замену?", _
                       vbOKCancel, "Подтверждение")
            If x = vbOK Then
               i.Value = T
               c = c + 1
            Else
               MsgBox "Отказ от замены"
            End If
         Else
            i.Value = T
            c = c + 1
         End If
      End If
   Next i
   MsgBox "Сделано " & Trim$(Str$(c)) & " замен(а/ы)"
Else
   MsgBox "Отказ от замены"
End If
```

Почти все описанные выше элементы управления могут быть вставлены не только в пользовательскую форму, но и прямо в документ (например, на рабочий лист Excel или на страницу документа Word), активизируя его. Это позволяет создать

удобные формы и шаблоны для ввода информации и ее обработки. Для этого используются инструменты панелей «Формы» и «Элементы управления».

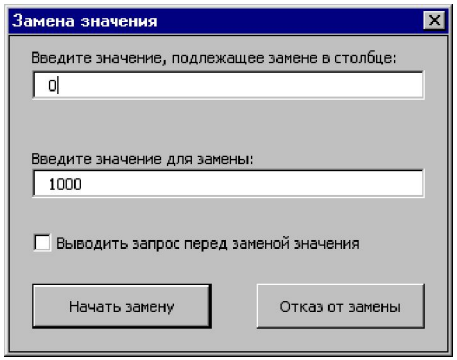


Рис. 6.4. **Пользовательская форма**

Возможности среды программирования VBA расширяются за счет применения дополнительных элементов управления. Файлы дополнительных элементов имеют обычно расширения VBX, OCX или DLL. Файлы VBX и OCX называют элементами управления ActiveX.

Дополнительный элемент управления – это многократно используемый компонент, являющийся самостоятельным объектом, реализованный обычно на C. Он имеет свои свойства и методы, как и любой другой объект.

Из программ VBA можно не только отображать пользовательские диалоговые окна, но и встроенные диалоговые окна. Для этого используется семейство ***Dialogs***. Это семейство содержится в объекте ***Application*** (рис. 6.5).

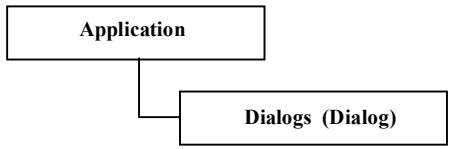


Рис. 6.5. **Семейство *Dialogs* объекта *Applications***

Объекты ***Dialog*** принадлежат семейству ***Dialogs***, которое содержит все встроенные диалоговые окна приложений MS Office. Можно создать новое диалоговое окно и добавить его в семейство. Для идентификации конкретного диалогового окна используется обращение ***Dialog***(**Индекс**), где индекс задается константой, имеющей префикс «***wdDialog***», за которым следует имя конкретного диалогового окна.

Более подробную информацию об этом классе можно получить, выполнив команду **Просмотр объектов** меню **Вид** редактора VBA. В списке «Классы» следует выделить данный элемент (***Dialogs*** или ***Dialog***) и в контекстном меню выбрать строку **Справка**. При вводе слова «Dialog» в коде программы редактор автоматически выдаст подсказку со списком используемых для выбора нужного окна констант (если включен режим подсказок). Например, вызов метода

```
dlgAnswer = Application.Dialogs(xlDialogPrint).Show
```

отображает встроенное диалоговое окно «Печать» приложения Excel. При вызове метода ***Show*** можно указывать аргументы для встроенных диалоговых окон.

6.13. Создание пользовательских меню и панелей инструментов

Меню в Microsoft Office, как и панели инструментов, могут перемещаться по экрану, «вставать» в определенные места, могут содержать кнопки с графическими изображениями и текстовые элементы.

Обобщением панелей инструментов и меню являются *панели команд*, которые программно реализованы на основе объекта ***CommandBar***. Эти панели могут выглядеть, как меню или как панели инструментов, но процессы их создания, изменения и управления ими осуществляются с помощью единых средств.

Семейство ***CommandBars*** содержится в объекте ***Application*** (рис. 6.6). Каждый объект ***CommandBar*** содержит, в свою очередь, семейство объектов ***CommandBarControl***. Эти объекты принадлежат одному из трех типов: ***CommandBarButton***, ***CommandBarComboBox*** или ***CommandBarPopUp***.

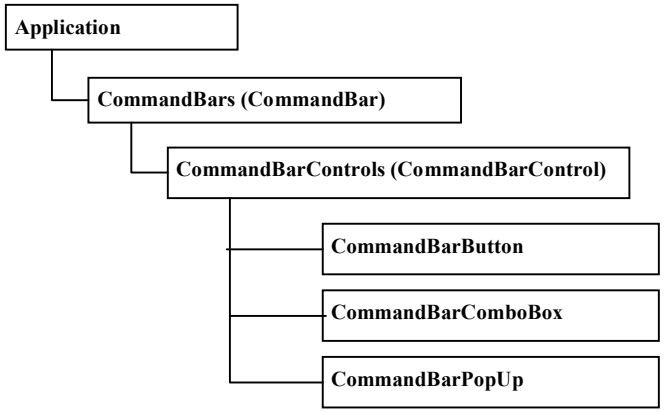


Рис. 6.6. **Иерархическая схема объекта *CommandBar***

Объект ***CommandBarButton*** представляет либо кнопку, либо элемент меню, которые выполняют команду или вызов подпрограммы.

Объект ***CommandBarComboBox*** представляет поле ввода, раскрывающийся список или поле со списком.

Объект ***CommandBarPopUp*** отображает меню и вложенные меню (подменю фактически представляют собой такой же объект ***CommandBar***, содержащийся в объекте ***CommandBarPopUp***).

Для идентификации конкретного объекта семейства используется ссылка вида ***CommandBars***(**Индекс**), где в качестве индекса выступает имя или номер, указывающие нужный объект, а в качестве результата возвращается ссылка на объект.

Для создания панели команд с помощью кода VBA можно использовать метод ***Add*** семейства объектов ***CommandBars***. Этот метод имеет следующие аргументы: **Name** – имя новой панели команд (может использоваться в качестве индекса для ее идентификации в будущем), **Position** – положение панели на экране (в качестве значения могут использоваться специальные константы, значения которых можно посмотреть в справочной системе при просмотре соответствующих объектов), **MenuBar** – значение типа ***Boolean*** (***True*** – панель превращается в активную строку меню, ***False*** – создается панель инструментов), **Temporary** – зна-

чение типа ***Boolean*** (***True*** означает, что панель удаляется после завершения приложения). Эта функция возвращает в качестве результата ссылку на объект ***CommandBar***.

Объект ***CommandBar*** имеет следующие *основные свойства*:

- ***Visible*** – панель является видимой, если это свойство установлено в ***True***; если значение свойства равно ***False***, панель является скрытой;
- ***Position*** – положение панели на экране, задается с помощью специальных констант;
- ***Controls*** – семейство объектов ***CommandBarControls***, элементы которого содержатся на панели команд.

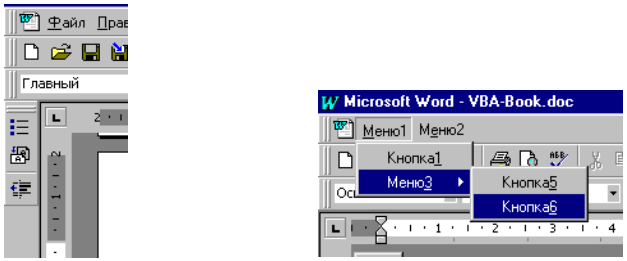
Манипулировать объектами ***CommandBar*** можно с помощью следующих *основных методов*:

- ***Delete*** – удаление панели команд (для встроенных панелей использоваться не может);
- ***Reset*** – восстановление исходного состояния встроенной панели команд;
- ***ShowPopUp*** – отображение контекстной pop-up панели команд.

Для создания нового объекта ***CommandBarControl*** можно использовать метод ***Add*** семейства объектов ***CommandBarControls***. Этот метод имеет следующие аргументы: **Type** – тип добавляемого объекта (задается константами: ***msoControlButton*** – кнопка или элемент меню, ***msoControlComboBox*** – поле со списком, ***msoControlDropdown*** – раскрывающийся список, ***msoControlEdit*** – поле ввода, ***msoControlPopup*** – вложенное меню); **ID** – используется для идентификации встроенного элемента управления (для элементов, определенных пользователем, остается пустым); **Parameter** – для элементов управления, определенных пользователем (обычно остается пустым); **Before** – индекс элемента управления, перед которым добавляется данный элемент (число или имя; если оставлен пустым, элемент добавляется в конец); **Temporary** – если это значение равно ***True***, элемент удаляется при закрытии приложения.

Объект ***CommandBarControl*** имеет следующие *основные свойства*:

- ***Caption*** – текстовая строка, отображаемая в заголовке (для создания горячей клавиши используется символ ‘&’);
- ***CommandBar*** – объект ***CommandBar***, который является вложенным меню элемента управления (используется только для ***CommandBarPopUp***);
- ***Enabled*** – если принимает значение ***True***, элемент управления доступен и может быть выбран пользователем, если ***False*** – элемент становится недоступным;
- ***FaceID*** – число, идентифицирующее стандартный встроенный рисунок, отображаемый на элементе управления;
- ***OnAction*** – имя подпрограммы VBA, которая будет выполнена при активизации элемента управления пользователем;
- ***Style*** – определяет, что будет отображено на кнопке: текст, рисунок или комбинация текста и рисунка (принимает значения констант ***msoButtonAutomatic***, ***msoButtonCaption***, ***msoButtonIcon***, ***msoButtonIconAndCaption***);
- ***ToolTipText*** – текст, отображаемый в окошке всплывающей подсказки элемента управления;
- ***Visible*** – если установлено в ***True***, элемент будет видимым, при значении ***False*** – скрытым.



а ***б***

Рис. 6.7. **Управление объектами *CommandBar*:**
***а*) созданная панель инструментов с тремя кнопками;**
***б*) созданное пользовательское меню**

Таким образом, появляется возможность управления панелями и меню не только «вручную», но и программно, с помощью кода, вставляемого в процедуры. Ниже приведены примеры таких процедур и результаты их выполнения.

Создание новой панели инструментов (результат выполнения процедуры приведен на рис. 6.7, *а*:

```
Sub AddToolBar ()
  With Application.CommandBars.Add ("МояПанель", , _
                                    False, True)
    .Visible = True
    .Position = msoBarLeft
    With .Controls
      With .Add(msoControlButton)
        .Caption = "Кнопка1"
        .FaceID = 12
        .OnAction = "Процедура1"
      End With
      With .Add(msoControlButton)
        .Caption = "Кнопка2"
        .FaceID = 13
        .OnAction = "Процедура2"
      End With
      With .Add(msoControlButton)
        .Caption = "Кнопка3"
        .FaceID = 14
        .BeginGroup = True
        .OnAction = "Процедура3"
      End With
    End With
  End With
End Sub
```

Кнопкам созданной панели приписаны подпрограммы **Процедура1**, **Процедура2** и **Процедура3**. Для проверки работоспособности приведенной выше подпрограммы можно использовать в качестве таких процедур заглушки вида:

```
Sub Процедура1 ()
  MsgBox "Нажата кнопка 1. Вызвана подпрограмма 1."
End Sub
```

При использовании в качестве значения свойства ***FaceID*** нуля кнопке приписывается пользовательская картинка (как значение свойства ***Picture***).

Создание нового меню:

```
Sub AddMenu ()
  With Application.CommandBars.Add ("НовоеМеню", , _
                                    True, True)
    .Visible = True
    With .Controls
      With .Add(msoControlPopUp)
        .Caption = "&Меню1"
        With .Controls
          With .Add(msoControlButton)
            .Caption = "Кнопка&1"
            .OnAction = "Процедура1"
          End With
          With .Add(msoControlPopUp)
            .Caption = "Меню&3"
            With .Controls
              With .Add(msoControlButton)
                .Caption = "Кнопка&5"
                .OnAction = "Процедура2"
              End With
              With .Add(msoControlButton)
                .Caption = "Кнопка&6"
                .OnAction = "Процедура3"
              End With
            End With
          End With
        End With
      End With
      With .Add(msoControlPopUp)
        .Caption = "М&еню2"
        With .Controls
          With .Add(msoControlButton)
            .Caption = "Кнопка&3"
            .OnAction = "Процедура4"
          End With
          With .Add(msoControlButton)
            .Caption = "Кнопка&4"
            .OnAction = "Процедура5"
          End With
        End With
      End With
    End With
  End With
End Sub
```

В данной процедуре создается новая панель меню с двумя пунктами, содержащая вложенные подменю (результат работы процедуры приведен на рис. 6.7, *б*).

Объекты ***CommandBar*** применяются для управления меню и панелями инструментов во время выполнения приложения. Ниже приведен код процедуры, демонстрирующей некоторые приемы управления: *запрещение*, *скрытие* и *удаление элементов управления*, а также восстановление меню.

```
Sub Ctrls()
     With Application.CommandBars("Menu Bar")
          With .Controls("&Таблица")
             .Enabled = False
             .Visible = True
          End With
          DoEvents
          MsgBox "Меню Таблица запрещено"
          With .Controls("&Таблица")
             .Enabled = True
             .Visible = False
          End With
          MsgBox "Меню Таблица скрыто"
          For Each C In .Controls
            C.Delete
            DoEvents
          Next C
          MsgBox "Удалены все меню"
          .Reset
          MsgBox "Строка меню восстановлена"
     End With
End Sub
```

Просмотреть имена всех видимых элементов управления можно с помощью следующей процедуры:

```
Sub LookCtrls()
    For Each C In Application.CommandBars
        If C.Visible Then
           MsgBox "Имя: " & C.Name
        End If
    Next C
End Sub
```

С помощью описанных выше средств на основе приложений Microsoft Office можно быстро разработать собственное приложение с удобным пользовательским интерфейсом.

6.14. Разработка контекстного меню

Методы создания строк меню и включения в них элементов управления можно применить к разработке контекстных меню. Для *построения контекстного меню* необходимо создать контекстное меню, добавить в него элементы управления и задать момент вывода меню.

Для *создания контекстного меню* используется метод ***Add*** семейства ***CommandBars***. В качестве значения параметра ***Position*** при вызове этого метода нужно указать константу ***msoBarPopup***. Метод возвращает ссылку на объект ***CommandBar***. В результате выполнения кода этого метода создается пустая строка меню.

Далее в контекстное меню можно *добавить элементы управления* любого типа (поле для ввода данных, раскрывающийся список, кнопку, запускающую процесс или задачу, подменю). Элементы управления также добавляются с помощью метода ***Add***, но добавление объектов, представляющих элементы управления, осуществляется для семейства ***Controls*** элементов управления созданного объекта ***CommandBar***. При вызове метода указывается тип создаваемого и включаемого в меню элемента управления.

Для включенных в контекстное меню элементов управления задаются свойства и действия (макросы), которые должны быть выполнены при активизации элементов.

Таким образом, процедура *создания контекстного меню* может выглядеть следующим образом:

```
Public Function СозданиеКонтекстногоМеню() As CommandBar
' Описание объекта для контекстного меню:
    Dim MyPopUp As CommandBar
' Описание элементов пункта меню "Формат":
    Dim cbFormatMenu As CommandBarPopup
' Описание элементов подменю "Цвет" меню "Формат":
    Dim cbFormatColorMenu As CommandBarPopup
    Dim cbRed As CommandBarButton
    Dim cbBlue As CommandBarButton
    Dim cbGreen As CommandBarButton
 ' Описание пункта "Размер" меню "Формат":
    Dim cbFormatSizeMenu As CommandBarComboBox
 ' Описание элементов пункта меню "Редактирование":
    Dim cbEditMenu As CommandBarPopup
    Dim cbDelete As CommandBarButton
```

```
    Dim cbCopy As CommandBarButton
    Dim cbInsert As CommandBarButton
    Set MyPopUp = CommandBars.Add("ShortcutMenu", _
                                  Position:=msoBarPopup)
' Создание объектов - элементов управления:
    With MyPopUp.Controls
       Set cbFormatMenu = .Add(Type:=msoControlPopup)
       cbFormatMenu.Caption = "Формат"
       Set cbFormatColorMenu = _
          cbFormatMenu.Controls.Add(Type:=msoControlPopup)
       cbFormatColorMenu.Caption = "Цвет"
       Set cbRed = _
                   cbFormatColorMenu.Controls.Add _
                   (Type:=msoControlButton)
       cbRed.Style = msoButtonCaption
       cbRed.Caption = "Красный"
       cbRed.OnAction = "SetRed"
       Set cbBlue = _
                   cbFormatColorMenu.Controls.Add _
                   (Type:=msoControlButton)
       cbBlue.Style = msoButtonCaption
       cbBlue.Caption = "Синий"
       cbBlue.OnAction = "SetBlue"
       Set cbGreen = _
                   cbFormatColorMenu.Controls.Add _
                   (Type:=msoControlButton)
       cbGreen.Style = msoButtonCaption
       cbGreen.Caption = "Зеленый"
       cbGreen.OnAction = "SetGreen"
       Set cbFormatSizeMenu = _
                   cbFormatMenu.Controls.Add _
                   (Type:=msoControlComboBox)
       cbFormatSizeMenu.Caption = "Размер"
       cbFormatSizeMenu.AddItem "8", 1
       cbFormatSizeMenu.AddItem "10", 2
       cbFormatSizeMenu.AddItem "14", 3
       cbFormatSizeMenu.ListIndex = 1
       cbFormatSizeMenu.OnAction = "SetSize"
       Set cbEditMenu = .Add(Type:=msoControlPopup)
       cbEditMenu.Caption = "Редактирование"
       Set cbDelete = _
          cbEditMenu.Controls.Add(Type:=msoControlButton)
       cbDelete.Style = msoButtonCaption
       cbDelete.Caption = "Удаление"
       cbDelete.OnAction = "DeleteBlock"
       Set cbCopy = _
          cbEditMenu.Controls.Add(Type:=msoControlButton)
```



```
       cbCopy.Style = msoButtonCaption
       cbCopy.Caption = "Копирование"
       cbCopy.OnAction = "CopyBlock"
       Set cbInsert = _
          cbEditMenu.Controls.Add(Type:=msoControlButton)
       cbInsert.Style = msoButtonCaption
       cbInsert.Caption = "Вставка"
       cbInsert.OnAction = "InsertBlock"
    End With
    Set СозданиеКонтекстногоМеню = MyPopUp
End Function
```

Отобразить контекстное меню можно с помощью метода ***ShowPopup*** объекта ***CommandBar***. Например, в следующей процедуре с помощью описанной выше функции создается и выводится на экран контекстное меню (переменная **MenuObj** описана как переменная ***Public*** типа ***CommandBar***):

```
Public Sub ShowMyMenu()
     Set MenuObj = СозданиеКонтекстногоМеню()
     MenuObj.ShowPopup
End Sub
```

Метод ***ShowPopup*** имеет необязательные параметры, задающие координаты вывода меню на экране.

При выполнении этой процедуры выведет на экран меню с двумя пунктами **Формат** и **Редактирование**. Каждому их пунктов соответствует свое подменю: при выборе пункта **Формат** раскрывается подменю из двух строк – **Цвет** и **Размер**, причем при выборе строки **Цвет** раскрывается еще одно подменю, содержащее команды **Красный**, **Синий** и **Зеленый**, а для ввода или выбора размера можно воспользоваться полем со списком значений «8», «10» и «14»; при выборе пункта меню **Редактирование** раскрывается подменю, содержащее три строки: **Удаление**, **Копирование** и **Вставка**. С пунктами меню **Красный**, **Синий**, **Зеленый**, **Размер**, **Удаление**, **Копирование** и **Вставка** связаны макросы с именами «SetRed», «SetBlue», «SetGreen», «SetSize», «DeleteBlock», «CopyBlock» и «InsertBlock» соответственно. Эти макросы выполняются при активизации пункта меню или изменении значения соответствующего элемента управления.

В макросе, связанном с элементом управления, можно по-

лучить свойства, установленные для выбранного элемента управления, и воспользоваться ими для выполнения соответствующей операции. Например, макрос

```
Public Sub SetSize()
   Dim CharSize As Single
   CharSize = Val(MenuObj.Controls(1).Controls(2).Text)
   MsgBox "Выбран размер " & _
                  MenuObj.Controls(1).Controls(2).Text
   Selection.Font.Size = CharSize
End Sub
```

выводит на экран выбранный размер и применяет его к выделенному тексту. Нужный элемент управления выбирается по его индексу в семействе объектов ***Controls***.

В приложении необходимо указать момент вызова процедуры создания и отображения меню. Этот момент может быть связан с определенным событием, связанным с объектом (например, щелчком правой кнопки мыши по нему). К сожалению, события, связанные с нажатием клавиш и щелчками мыши, поддерживаются не всеми приложениями.

6.15. Открытие проектов, экспорт и импорт

Любой проект связан с некоторым документом или шаблоном основного приложения. Следовательно, для просмотра проекта и работы с ним требуется открыть связанный с ним документ.

Каждое приложение имеет свой способ хранения проектов:

- Word сохраняет проекты в документах (файлах с расширением DOC) и шаблонах документов (файлах DOT);
- Excel сохраняет проекты в рабочих книгах (XLS);
- Access сохраняет проекты в файлах базы данных (MDB);
- PowerPoint сохраняет проекты в презентациях (PPT).

Проекты являются совокупностью модулей. Каждый проект может содержать как *стандартные модули* (с описаниями и кодом процедур SUB и FUNCTION, которые могут быть доступны во всем проекте), так и *модули классов* (содержат описание объекта, который является членом соответствующего класса; процедуры, описанные в модуле класса используются только в этом модуле).

Редактор VBA обеспечивает возможность импорта и экспорта компонентов приложения, что позволяет использовать имеющиеся в одном приложении данные в других приложениях и проектах.

Если имеется компонент проекта VBA, который необходимо включить в другой проект, то сначала необходимо его экспортировать из исходного проекта, а затем импортировать в разрабатываемый проект. Кроме того, так как компоненты приложений VBA всегда хранятся в отдельных файлах, можно загрузить компонент проекта.

Каждый компонент приложения VBA хранится в основном документе, поэтому, чтобы использовать их проекте VBA в другом документе или приложении, следует сначала выполнить экспорт.

Чтобы *экспортировать файл* нужно:

1. Выбрать экспортируемый компонент в окне проекта.
2. Выполнить команду **Экспорт файла** в контекстном меню или в меню **Файл**.
3. В диалоговом окне команды раскрыть нужную папку и ввести имя файла для сохранения в нем экспортируемого компонента.
4. Щелкнуть кнопку **OK**.

Для файлов VBA используются следующие расширения: FRM – форма, CLS – класс, BAS – модуль программы.

Чтобы *импортировать файл* в проект следует:

1. В окне редактора VBA (в окне проекта) выполнить команду **Импорт файла** с помощью контекстного меню или меню **Файл**.
2. Найти и открыть нужный файл.

Вопросы для самопроверки

Запуск программы и работа в редакторе VBA

1. Каким образом можно вызвать редактор VBA?
2. Какими способами можно запустить на выполнение макрос?
3. Какие ключи можно указать при открытии документа? Каково действие этих ключей?

Структура программы, понятие подпрограммы и функции

4. Какова структура программ, написанных на VBA? Какие элементы включают программы?
5. Что такое процедура, подпрограмма и функция? Для чего выделяются подпрограммы, процедуры и функции в тексте программ? В чем их отличие?
6. Каковы основные правила записи процедур, подпрограмм и функций в VBA?
7. Что такое параметры? Какими способами можно передать аргументы в процедуру? Каковы правила передачи? Приведите примеры передачи параметров различными способами. Каковы преимущества и недостатки каждого способа?
8. Какие процедуры называются рекурсивными? Покажите, как можно определить рекурсивный алгоритм вычисления факториала, Какие еще рекурсивные алгоритмы вы моете привести?
9. Какие стандартные функции VBA вы знаете? Приведите примеры их использования.

Типы данных и описания в VBA

10. Какие типы данных используются в VBA?
11. Определите, что такое переменная? В чем отличие переменных и констант? Приведите примеры переменных и констант.

12. Каковы правила записи имен переменных в VBA? Приведите примеры правильно записанных имен и имен, в которых имеются ошибки.
13. Каким образом определяется тип переменных в VBA? Приведите примеры ошибок, связанных с описанием переменных, поясните примеры.
14. Как описать переменные? Какова область действия описаний?
15. Что такое массив? Как описываются массивы? Приведите пример описания и использования массива.
16. В чем отличие массивов фиксированного и переменного размера? Приведите примеры динамических массивов.
17. Приведите примеры одно- и двумерных массивов различных типов.
18. Что такое объект? Определите понятие объектной переменной.
19. Каким образом можно обратиться к свойству или методу объекта?
20. Может ли пользователь определить собственный тип?
21. Когда требуется выполнять преобразование типов? Приведите примеры.
22. Какие возможности для преобразования типов имеются в распоряжении программистов, разрабатывающих программы на VBA? Приведите примеры их использования.

Выражения в VBA

23. Какие группы операций можно использовать в выражениях? Приведите примеры операций, относящихся к различным группам.
24. Что определяет приоритет операции? Перечислите операции в порядке убывания приоритетов.
25. Как определяется тип результата вычисления выражения? Приведите примеры правильно записанных выражений и укажите типы результатов для них.
26. Приведите примеры выражений, в записи которых сделаны ошибки. Поясните их.

Управляющие структуры и события в программах на VBA

27. Определите понятие события. Приведите примеры.
28. Перечислите управляющие структуры, используемые в программах на VBA.
29. Какие варианты конструкции ветвления используются в программах на VBA? Опишите синтаксические правила для них. Приведите примеры их использования в вычислениях.
30. Какие варианты конструкции повторения используются в программах на VBA? Опишите синтаксические правила для них. Приведите примеры использования циклов.
31. Что может быть параметром цикла в программе на VBA? Приведите примеры.
32. Какого типа могут быть счетчики в циклах? Как меняется значение счетчика в программе на VBA? Приведите примеры.
33. Можно ли в программе на VBA «досрочно» выйти из цикла? Можно ли выйти из цикла с предусловием, если в качестве условия, управляющего циклом, указано логические значение истины?
34. Какой вид цикла нужно использовать, если требуется последовательно открыть все файлы MS Word, находящиеся в заданном каталоге (папке), если папка может быть пустой?
35. Какой тип цикла нужно использовать, если требуется последовательно выводить на экран содержимое ячеек столбца таблицы Excel, пока не будет найдено заданное значение (при условии, что оно обязательно содержится в одной из ячеек)?
36. Какой тип цикла лучше использовать в программе для перебора и удаления всех рисунков, включенных в текст?
37. Какой тип массива нужно использовать, чтобы просуммировать значения элементов массива с четными номерами (значениями индексов)?

Проектирование и разработка интерфейса

38. Какие средства диалога с пользователем, ввода и вывода данных имеются в VBA?
39. Какие элементы управления можно использовать при разработке форм (диалоговых окон)? Опишите их. В каких случаях применяются эти элементы? Приведите примеры.
40. Какие средства для создания меню и панелей инструментов имеются в VBA?
41. Имеется ли возможность создать пользовательское контекстное меню?

Проекты: открытие, экспорт и импорт

42. Определите понятие проекта в VBA.
43. Как создаются и хранятся проекты в приложениях MS Office?
44. Что такое модуль? Какие типы модулей вы знаете?
45. Какие типы файлов применяются при экспорте и импорте проектов, модулей?

Обработка ошибок

46. Какие возможности для обработки ошибок имеются в программах на VBA?
47. Перечислите средства обработки ошибок.
48. Приведите примеры обработки ошибок.

Глава 7. Использование Win32 API

Win32 Application Programming Interface (API) – библиотека, доступная каждому программисту, создающему приложения в среде Windows. Она используется для создания приложений для любых 32-разрядных операционных систем фирмы Microsoft.

При разработке приложений требуются средства формализации процесса повторного использования кода, с помощью которых разные разработчики могли бы использовать одни и те же коды программ.

В Microsoft Office шаблоны являются каркасами приложений, примером практической реализации принципа многократного использования программных компонентов.

Операционные системы Windows разрабатывались на основе модели WOSA (Windows Open Services Architecture – открытая архитектура служб Windows). В соответствии с этой концепцией API служит своего рода черным ящиком: все функции, используемые в Windows, вынесены «на всеобщее обозрение» (известны их имена, назначение, списки параметров, возвращаемые значения), но без описания деталей их реализации.

Все сервисы WOSA делятся на три группы:

- *Базовые сервисы*: Win32 API (позволяет использовать в приложениях практически любые функции Windows 95 или Windows NT); ODBC API (возможность связываться с любыми источниками данных независимо от разработчика); MAPI (Messaging API – реализация доступа к почтовым системам, использующим MAPI); TAPI (Telephony API – доступ к коммутируемым (телефонным) линиям); OLE API (поддержка связывания и внедрения объектов);
- *Коммуникационные сервисы*: Sockets API (доступ к сети с помощью различных сетевых протоколов); SNA API (открытый доступ к большим ЭВМ – мэйнфреймам фирмы IBM); RPC API (удаленный вызов процедур, позволяющий упростить разработку клиент/серверных распределенных систем);

- *Сервисы вертикальных приложений (бизнес-сервисы)* – расширения WOSA для систем электронной коммерции, обработки биржевой информации в реальном времени, для систем управления, разработки и производства.

Можно сказать, что Win32 API состоит из двух компонентов: библиотек динамической компоновки (DLL) и описаний к ним. Создание нового API сводится фактически к разработке новой библиотеки.

Использование функций Win32 API расширяет возможности VBA. Для подключения функций Win32 API их необходимо описать с помощью специальной инструкции ***Declare***. Эта инструкция (описатель) сообщает программе, где находится требуемая функция API и как ее использовать. В описателе указывается следующая информация: *имя функции*, *содержащая функцию DLL*, *передаваемые в функцию параметры*, *тип возвращаемых данных*. Правило записи инструкции выглядит следующим образом:

```
Declare Function | Sub Псевдоним Lib "ИмяБиблиотеки" _
    [Alias "ИмяФункцииAPI"](СписокПараметров) _
    [As ТипРезультата]
```

Основные библиотеки Windows 95 имеют следующие назначения: KERNEL32 – *функции ядра* ОС (низкоуровневые функции управления ресурсами и процессами), GDI32 – *функции управления графическим интерфейсом*, USER32 – *функции для работы с окнами*.

Если функция API возвращает данные строкового типа, необходимо указать буфер, представляющий собой переменную для хранения возвращаемой строки, и число, определяющее длину буфера. Строки, возвращаемые функциями API, заканчиваются символом с кодом 0 (это ASCIIZ-код), поэтому размер буфера должен быть на 1 больше предполагаемой длины строки.

При передаче в функцию API строковой переменной она воспринимается как указатель. Для передачи в функцию пустого указателя в VBA используется значение 0& (нуль типа ***Long***). При передаче в функцию пустого указателя необходимо внести в описатель функции изменения: тип аргумента можно указать

как ***As Any***, при этом в функцию можно передать как строковые переменные, так и переменные типа ***Long***, если нужно.

Для облегчения работы с объектами в Windows используются их числовые идентификаторы. Существует два типа таких идентификаторов: дескрипторы и указатели. *Дескриптор* – это специальное 32-битное целое число, которое используется Windows для идентификации объекта, например, окна или потока. Большинство функций Win32 API работает с дескрипторами. *Указатель* – это адрес объекта, переменной или структуры. VBA их не использует, но многие функции API требуют передачи указателей в качестве аргументов. При вызове данных функций из программы VBA необходимо использовать переменную того же типа, что и переменная, на которую ссылается указатель, и использовать ключевое слово ***ByRef*** в описании функции.

Текстовые строки всегда передаются в функции из библиотек DLL по ссылке. Ключевое слово ***ByVal*** для строковых аргументов задает преобразование строки в формат с окончанием на нулевой символ. Использование ключевого слова ***ByRef*** для строковых аргументов недопустимо.

Ниже приведен пример использования функции GetSystemDirectory для получения полного имени системной папки Windows. В модуле, содержащем вызов функции, должен содержаться описатель:

```
Declare Function GetSystemDirectory Lib "KERNEL32"  _
          Alias "GetSystemDirectoryA"  _
          (ByVal lpBuffer As String ,  _
          ByVal nSize As Long) As Long
```

Далее приведен пример процедуры, обращающейся к данной функции:

```
Sub Ex_GetSystemDirectory ()
   Dim strBuffer As String
   Dim intLen As Integer
' Выделение памяти для буфера (возвращаемой строки):
   strBuffer = String(256, 0)
' Получение полного имени системной папки:
   intLen = GetSystemDirectory (strBuffer, _
                                Len(strBuffer))
' Отсекается пустой «хвост» строки:
   strBuffer = Left(strBuffer, intLen)
   MsgBox Prompt := strBuffer, _
          Buttons := vbInformation, _
          Title := "Системная папка"
End Sub
```


Свойство ***LastDLLError*** возвращает *системный код ошибки*, возникшей при вызове библиотеки динамической компоновки (DLL). Это свойство доступно только для чтения, применимо только при вызовах в программах Visual Basic библиотек динамической компоновки (DLL). После вызова библиотеки вызываемая функция обычно возвращает код, свидетельствующий об успешном или неудачном выполнении, и заполняет значение свойства ***LastDLLError***. Для того чтобы получить список возвращаемых значений и их описание, следует обратиться к документации по функциям DLL. В любом случае возвращения кода неудачного вызова в приложение, использующее Visual Basic, необходимо немедленно проверить значение свойства ***LastDLLError***.

Псевдонимы целесообразно использовать в некоторых случаях для вызова функций API. Например:

- При использовании функций API с именами, в которых присутствуют символы, недопустимые для использования в названиях функций VBA.
- При использовании пользовательских нестандартных описателей.
- При использовании функций работы со строками, чтобы подчеркнуть, когда используется набор символов ANSI (тогда к имени функции добавляется суффикс ‘A’), а когда – Unicode (добавляется суффикс ‘W’).

Некоторые функции Win32 API используют *стандартные константы*. Использование имен этих констант, а не конкретных значений, гарантирует правильность работы программы при изменениях в Win32 API. Приложение API Viewer, входящее в комплект поставки Microsoft Office, позволяет скопировать в модуль VBA описания подобных констант.

В именах параметров функций Win32 API используются *стандартные префиксы*, принятые в C/C++:

- ***lp*** – указатель на переменную, обычно на строку (в VBA используется строковая переменная);
- ***n*** – число (в VBA используется переменная типа ***Integer***);

– ***dw*** – двойное слово без знака (в VBA используется тип ***Long***);
– ***w*** – слово без знака (в VBA используется тип ***Integer***);
– ***h*** – дескриптор (в VBA используется тип ***Long***);
– ***u*** – целое без знака (в VBA используется переменная типа ***Integer***).

При установлении соответствия между типами, используемыми в C/C++ и типами, поддерживаемыми в VBA, могут производиться преобразования.

Вопросы для самопроверки

1. Что такое API?
2. Каково назначение API? Когда требуется использовать функции API при работе в VBA?
3. Перечислите группы сервисов WOSA.
4. Приведите синтаксис описания функций API для подключения их к приложениям VBA.
5. Напишите пример процедуры, обращающейся к функции API.
6. Каково назначение псевдонимов?
7. Какие префиксы используются в именах параметров функций API? Каким типам они соответствуют?
8. Какие стандартные константы, используемые при вызовах функций API, вы знаете?

Глава 8. Использование шаблонов и надстроек

Все продукты MS Office позволяют сохранять приложения в отдельных документах, но этот вариант плохо подходит для многократного использования. Задачу многократного применения приложений решают шаблоны и надстройки. Шаблоны решают, в частности, задачу разделения кода.

Шаблоны упрощают создание и форматирование документов с общим содержанием и общей функциональностью, задают модель приложения. Шаблоны могут включать текст, графику, формулы, специальные элементы оформления текста, таблиц, рисунков и т.п., диалоговые окна пользователя, модули VBA.

Каждый продукт использует специальное расширение для файлов шаблонов: DOT – для Word, XLT – для Excel, POT – для PowerPoint.

Чтобы шаблон автоматически загружался каждый раз при запуске приложения Office, его нужно сохранить в папке «XLSTART” (для Excel) или «STARTUP» (в случае Word или PowerPoint).

Все документы Word используют совместно шаблон, на котором они основаны. Практически это означает, что документы Word состоят из двух проектов: первый проект от используемого шаблона, а второй – собственно документ Word.

Рабочие книги Excel и презентации PowerPoint, основанные на шаблонах, используют индивидуальные копии исходного шаблона. Они могут содержать только один проект VBA: каждый файл содержит свою копию шаблона, поэтому изменения, сделанные в этой копии не отражаются на других документах.

Шаблоны Word могут включать следующие элементы: автотекст, закладки, комбинации клавиш, стили, текст и другие объекты (поля, гиперссылки, разделы, колонтитулы и надписи), панели инструментов и меню, модули и формы. Они сохраняются обычно в папке «Шаблоны».

На документ Word могут влиять несколько шаблонов, причем шаблон, на котором непосредственно основан документ, может содержать один набор установок, а глобальный шаблон NORMAL.DOT – другой. Кроме того, могут быть загружены и другие глобальные шаблоны. Разрешение конфликтов основывается на иерархии шаблонов. В зависимости от характера связи шаблона с документом, установки шаблона могут отменять или не отменять установки другого шаблона для данного документа. Иерархия шаблонов выглядит следующим образом:

– шаблон, присоединенный к активному документу;
– шаблон «Обычный»;
– другие глобальные шаблоны;
– надстройки;
– уровень приложения.

Установки, сделанные в шаблоне, присоединенном к документу, имеют максимальный приоритет.

Модули VBA и диалоговые окна обычно сохраняются в глобальных (общих) шаблонах, доступных всем документам Word (примером является шаблон NORMAL.DOT или «Обычный»). Чтобы сделать шаблон глобальным, нужно воспользоваться командой **Шаблоны и надстройки** из меню **Сервис**.

Если шаблон сохранить в папке «STARTUP», то Word будет автоматически использовать этот шаблон при запуске или при нажатии кнопки **Создать** на панели инструментов.

В Excel можно использовать три категории шаблонов: шаблоны, автоматизирующие решение конкретных задач; автошаблоны, которые имеют конкретное имя и могут быть рабочими листами или рабочими книгами; пользовательские шаблоны листов, которые можно использовать в качестве основы для новых рабочих листов, листов диаграмм, листов с модулями, диалоговыми формами и макросами.

Excel распознает следующие имена автошаблонов: BOOK.XLT (для рабочих книг), CHART.XLT (для листов диаграмм), SHEET.XLT (для рабочих листов), DIALOG.XLT (для диалоговых форм Excel версий 5 и 7), MODULE.XLT (для модулей с кодом Excel версий 5 и 7), MACRO.XLT (для листов макросов Excel версий 4).

Для создания автошаблона или пользовательского шаблона рабочего листа нужно удалить все листы, которые не будут использоваться в качестве шаблона, оставив только необходимые таблицы, диаграммы, формы, модули и макросы.

Если шаблон BOOK.XLT сохранить в папке «XLSTART», то Excel при запуске или при нажатии кнопки **Создать** на панели инструментов приложения будет автоматически использовать этот шаблон для создания новой книги, а если в этой папке сохранить автошаблон листа, то он будет использоваться всякий раз при вставке в рабочую книгу листа такого типа.

Шаблоны PowerPoint позволяют сохранять форматирование, графику и модули VBA. Для обеспечения функциональности пользовательского интерфейса необходимо использовать надстройки PowerPoint.

В Access шаблоны играют меньшую роль, чем в других продуктах MS Office. Эти шаблоны могут содержать только простейшие элементы оформления форм и отчетов.

Для решения задачи обновления приложений Excel и PowerPoint можно применять *надстройки*, представляющие собой автономные программные компоненты, которые могут использоваться как приложениями, разработанными средствами Microsoft Office, так и самими продуктами Microsoft Office. При этом всегда используется один и тот же код, защищенный от локальных изменений внутри приложения.

В каждом продукте MS Office для файлов надстроек используется собственное расширение: XLA – для Excel, PPA – для PowerPoint, MDA – для Access.

Для создания надстройки в Excel или PowerPoint следует сохранить файл с особым расширением, указывающим, что в нем содержится надстройка (команда **Сохранить как…** меню **Файл**). Чтобы надстройка автоматически загружалась во время запуска приложения, ее нужно поместить в папку «XLSTART» (для Excel) или «STARTUP» (для PowerPoint). Надстройки можно загружать вручную, используя команду **Надстройки** из меню **Сервис**. Возможна программная загрузка из процедуры модуля VBA.

В Access для создания надстроек используется команда **Служебные программы ▸ Создать MDE-файл** из меню **Сер-**

вис. Эта команда дает возможность сохранить файл как надстройку.

В Word не поддерживается работа с надстройками в той же форме, что и в других продуктах Office. Фактически надстройки Word являются глобальными шаблонами или библиотеками WLL. Для загрузки надстройки Word соответствующий файл можно поместить в папку «» (если необходима автоматическая загрузка во время запуска). «Вручную» надстройку можно загрузить, выполнив команду **Шаблоны и надстройки** меню **Сервис**.

Использование надстроек способствует *защите приложения* от изменений. После создания надстройки невозможно редактировать или просматривать исходный код, поэтому перед ее созданием следует убедиться, что исходный файл шаблона или базы данных был сохранен (как «резервная копия»). Надстройки не могут содержать текста документа, формул и форматирования (например, именованных диапазонов Excel, закладок Word).

Вопросы для самопроверки

1. Определите понятие шаблона.
2. Какие возможности предоставляют пользователю шаблоны Word?
3. Какие средства можно использовать при создании шаблона Excel?
4. Какие расширения используются для файлов шаблонов в Microsoft Office?
5. Какие шаблоны используются при создании документов Word, какова их связь?
6. Каковы особенности размещения и использования шаблонов в Excel?
7. Что такое надстройка и в чем ее отличие от шаблона?
8. Каков порядок создания надстройки?
9. Каким образом используются надстройки в приложениях MS Offoce?

Глава 9. Разработка приложений с помощью Excel

Компонент Excel является наиболее популярным для разработки приложений, предназначенных для различных сфер: клиентских приложений для баз данных; финансовых моделей и инструментов анализа; систем подготовки отчетов, статистических и финансовых презентаций; экспертных информационных систем.

9.1. Иерархия объектов Excel

Объектная модель Excel содержит свыше 100 встроенных объектов и объекты, совместно используемые всеми приложениями Microsoft Office. К совместно используемым относятся ***CommandBars Data Access Objects*** (объекты доступа к данным), ***Forms*** (пользовательские диалоговые окна и элементы управления), ***Office Assistant*** и ***FileSearch***. Объекты из других приложений Microsoft Office и BackOffice можно также использовать при разработке приложений на основе Excel.

Объекты Excel позволяют осуществлять анализ данных для широкого класса задач. Приложения создаются путем объединения объектов с помощью VBA. Все объекты подробно документированы в справочной системе VBA.

Объекты Excel находятся на разных уровнях иерархии. Для их эффективного использования в программах VBA необходимо представлять себе эту иерархию (рис. 9.1). На рис. 9.2-9.3 показана иерархическая схема объекта ***WorkBook***. Символ ‘ ▸ ’ справа от названия объекта означает, что его структура раскрыта на других схемах.

Расположенный на верхнем уровне иерархии объект ***Application*** представляет Microsoft Excel. Остальные объекты расположены на следующих уровнях иерархии.

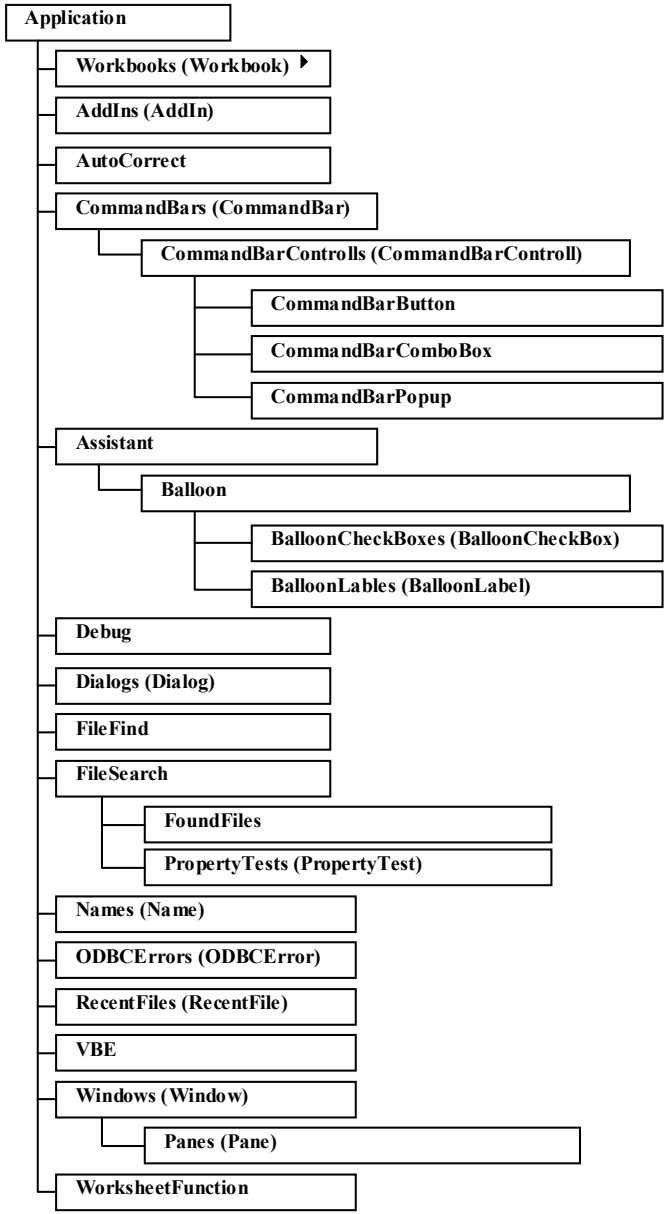


Рис. 9.1. **Иерархия объектов Excel**

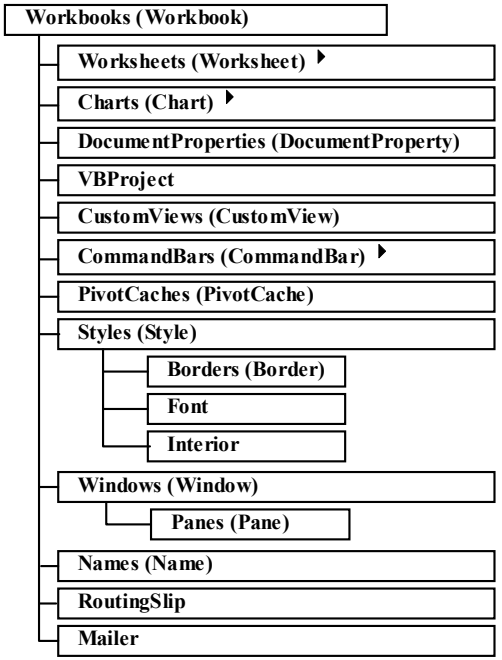


Рис. 9.2. **Иерархическая структура объекта** *Workbook*

Второй уровень включает объекты:

AddIn – объект, представляющий файл надстройки Excel;

Assistant – объект для управления помощником (совместно используется всеми приложениями MS Office);

AutoCorrect – объект для доступа к средствам автозамены;

CommandBar – объект, представляющий панели команд;

Debug – объект, представляющий окно отладки;

Dialog – объект, представляющий встроенное диалоговое окно;

FileFind – объект для доступа к средствам диалогового окна «File Find» (только для Macintosh);

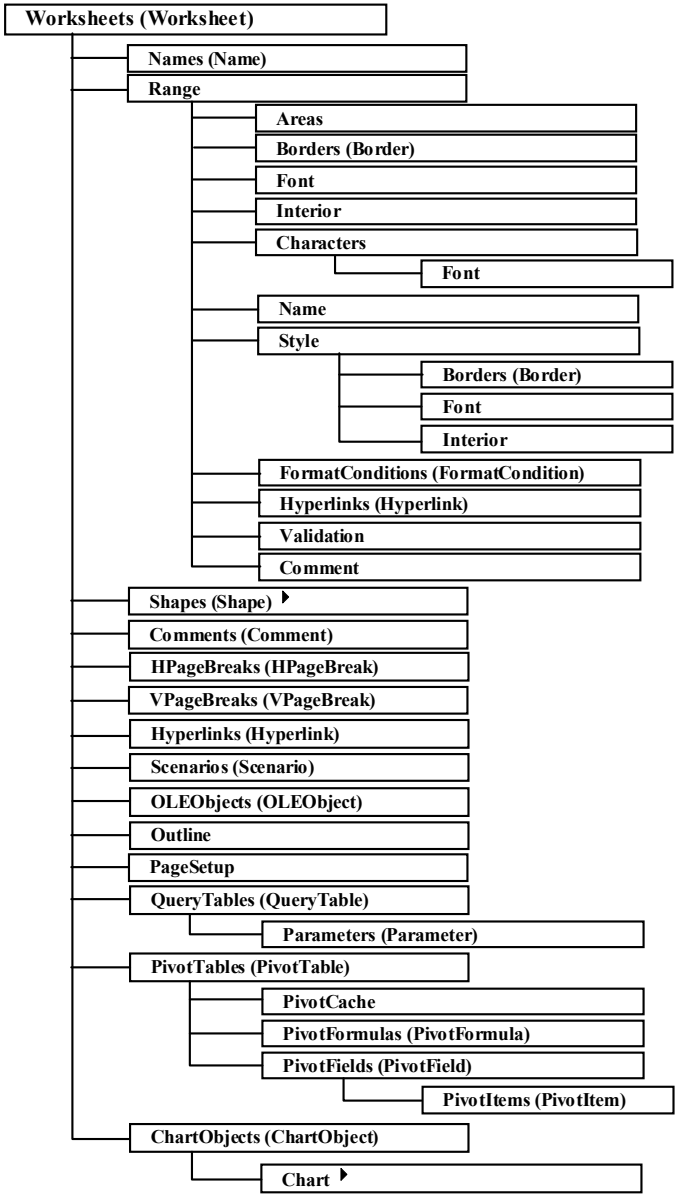


Рис. 9.3. **Иерархическая схема объекта** *Worksheet*

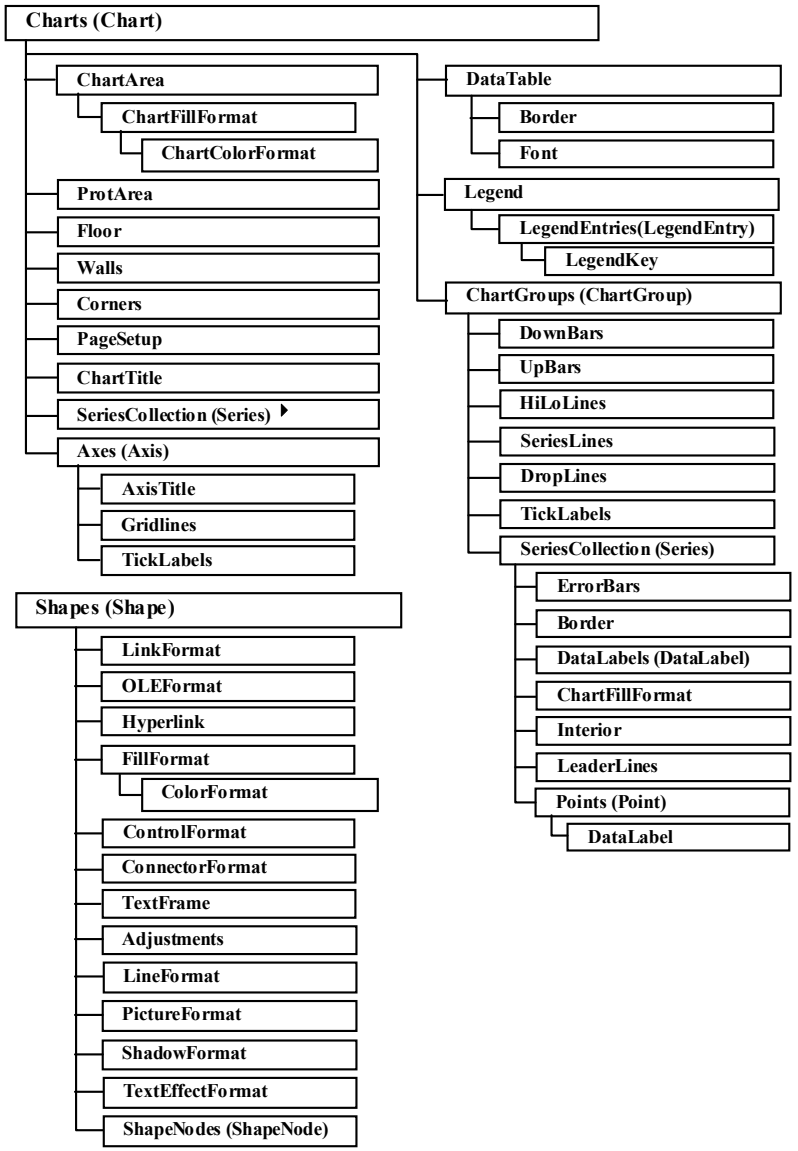


Рис. 9.4. **Иерархическая схема объектов** *Chart* **и** *Shape*

FileSearch – объект, применяемый для поиска файлов (совместно используется приложениями Office);

Name – объект, представляющий имя диапазона ячеек;

ODBCError – объект, используемый для представления ошибок при операциях с базами данных через ODBC;

RecentFile – объект, представляющий файл из списка последних использовавшихся файлов;

VBE – объект для управления редактором VBA (совместно используется приложениями Office);

Window – объект, используемый для доступа к окнам;

Workbook – объект, представляющий файл рабочей книги;

WorkSheetFunction – объект, используемый для исполнения функций рабочего листа из программы на VBA.

Следующие уровни иерархии включают набор дополнительных объектов, используемых для поддержки (обеспечения функционирования) объектов второго уровня. На рис. 9.1-9.4 изображены основные объекты Excel. В приведенные схемы не включены некоторые объекты, оставленные для совместимости с предыдущими версиями, объекты Microsoft Map и др. Более полную информацию можно найти в диалоговом окне «Просмотр объектов» и в справочной системе VBA.

Чтобы управлять свойствами и методами объекта, иногда нужно записать ссылки на все объекты, находящиеся в иерархии выше него. Ссылки на объекты записываются последовательно, начиная с верхнего уровня иерархии. В качестве разделителя между ссылками (именами) ставится точка. Например, оператор

```
Application.Workbooks("Книга1").Worksheets(1). _
                                Range("A1").Value = 1
```

заносит значение в указанный диапазон ячеек заданного рабочего листа выбранной рабочей книги.

Ссылки на объекты в семействе требуют указания индексов.

Для сокращения записи можно воспользоваться оператором ***With***.

В некоторых случаях нет необходимости в перечислении всех ступеней иерархии. С какого уровня можно начать ссылки, зависит от контекста вызова метода или обращения к свойству.

Если код выполняется из среды Excel, то указание объекта ***Application*** не является необходимым.

Для неявной ссылки на рабочую книгу можно использовать ссылки ***ActiveWorkbook*** (ссылка на активную рабочую книгу, с которой в данный момент работает пользователь) и ***ThisWorkbook*** (ссылка на рабочую книгу, в которой хранится исполняемый в настоящий момент код). Если открыта только одна книга, то ссылку на рабочую книгу можно опустить. Если открыто насколько рабочих книг, а ссылка на книгу опущена, действия выполняются над активной рабочей книгой.

Для доступа к активному рабочему листу можно использовать ссылку ***ActiveSheet***. Ссылка на активный лист также может быть опущена.

На активный (т.е. выделенный) диапазон ячеек можно сослаться, используя свойство ***Selection*** объекта ***Application***. Но это свойство может ссылаться и на другие объекты помимо ячеек, поэтому при использовании такой ссылки не следует полагаться на использование свойства по умолчанию.

Ссылки ***ActiveWorkbook***, ***ThisWorkbook***, ***ActiveSheet*** и ***Selection*** – это ссылки на свойства объектов.

9.2. Обзор основных объектов Excel

Ниже рассматриваются только основные, наиболее часто используемые объекты Excel. Более полную информацию можно получить в справочной системе. Удобным средством является браузер объектов, который позволяет найти по имени нужный объект или его свойства, методы.

Для поиска наименований нужных объектов, их свойств и методов можно воспользоваться также макрогенератором, выполнив нужные действия «вручную» и записав их последовательность в макрос.

9.2.1. Объект Application

Объект ***Application*** является главным в иерархии. Его можно представить себе как среду, в которой выполняется приложение.

Чаще всего используются следующие *свойства* объекта:

- ***ActiveWorkbook***, ***ActiveSheet***, ***ActiveCell*** – ссылки на активные объекты приложения;
- ***Caption*** – надпись, отображаемая в заголовке окна Excel (тип – ***String***; доступно для чтения и записи);
- ***Cursor*** – установка указателя мыши (***xlNormal***, ***xlDefault*** – обычный указатель (по умолчанию), установленный в Панели управления Windows, ***xlWait*** – песочные часы, ***xlNorthWestArrow*** – стрелка вверх и влево, ***xlBeam*** – I-курсор, как при редактировании текста);
- ***DisplayAlerts*** – управляет выводом на экран встроенных предупреждений Excel (тип – ***Boolean***; доступно для чтения и записи);
- ***DisplayStatusBar*** – управляет отображением строки состояния (тип – ***Boolean***; доступно для чтения и записи);
- ***DisplayFormulaBar*** – управляет отображением строки формул (тип – ***Boolean***; доступно для чтения и записи);
- ***DisplayScrollBars*** – управляет отображением полос прокрутки (тип – ***Boolean***; доступно для чтения и записи);
- ***EnableCancelKey*** – обычно прерывание выполнения макроса в приложении осуществляется с помощью комбинации клавиш *Ctrl+Break*; в том случае, когда требуется запретить прерывание выполнения макроса, данному свойству нужно присвоить значение константы ***xlDisabled*** (данная установка позволяет выполнить макрос как единую транзакцию, но этим значением следует пользоваться очень осторожно – нужно убедиться, что макрос не может зациклиться), а в том случае, когда после прерывания макроса следует передать управление на процедуру обработки ошибок, нужно установить для данного свойства значение ***xlErrorHandler*** (в этом случае в макрос должна быть включена последовательность операторов для обработки ошибок, которая указывается с помощью

инструкции ***On Error GoTo***, а вырабатываемый в данном случае код ошибки равен 18);
- ***OperatingSystem*** – строка текста с информацией об операционной системе (только для чтения);
- ***Path*** – путь к каталогу (папке), в котором установлен Excel (тип – ***String***; только чтение);
- ***ScreenUpdating*** – управляет обновлением экрана при выполнении подпрограммы (тип ***Boolean***; доступно для чтения и записи; устанавливается только на время выполнения процедуры, в которой сделана установка);
- ***StatusBar*** – текст строки состояния;
- ***ThisWorkBook*** – ссылка на объект ***WorkBook*** – рабочую книгу, в которой содержится выполняемый макрос;
- ***UserName*** – имя текущего пользователя (по умолчанию равно значению поля «Имя пользователя» на вкладке «Общие» диалогового окна, открываемого командой **Параметры** из меню **Сервис**);
- ***Version*** – номер версии Excel (только для чтения);
- ***WindowState*** – состояние окна приложения (может принимать три значения: ***xlNormal*** – нормальное, ***xlMaximized*** – развернуто на весь экран, ***xlMinimized*** – свернуто; доступно для чтения и записи).

Основные *методы* данного объекта:

- ***Calculate*** – вызывает перевычисление всех формул на всех рабочих листах во всех открытых рабочих книгах;
- ***GoTo*** – выбор требуемого диапазона или процедуры VBA в любой рабочей книге;
- ***InputBox*** – используется для получения информации от пользователя;
- ***OnKey***, ***OnRepeat***, ***OnTime***, ***OnUndo*** – методы, вызываемые обычно в процедуре ***Auto_Open*** модуля «AutoExec», эти методы позволяют задать реакцию приложения на определенные события:
  - ***OnKey*** – задает процедуру, которая должна быть выполнена при нажатии определенной клавиши или комбинации клавиш (коды клавиш и имя процедуры передаются при вызове метода как строковые аргументы);

- ***OnRepeat*** – задает процедуру, которая должна быть вызвана при выполнении команды **Повторить** ***действие*** из меню ***Правка*** (полное название команды и имя процедуры передаются при вызове метода как строковые аргументы);
- ***OnUndo*** – задает процедуру, которая должна быть вызвана при выполнении команды **Отменить** ***действие*** из меню ***Правка*** (полное название команды и имя процедуры передаются при вызове метода как строковые аргументы);
- ***OnTime*** – позволяет назначить выполнение процедуры на заданное время;

– ***Quit*** – закрывает Excel;
– ***Run*** – запускает макрос или подпрограмму.

В программах на VBA можно обрабатывать следующие *события объекта* ***Application***:

– ***NewWorkbook*** – создание новой рабочей книги;
– ***WorkbookActivate*** – активизация любой рабочей книги;
– ***WorkbookAddinInstall*** – установка рабочей книги в качестве дополнительной надстройки;
– ***WorkbookAddinUninstall*** – отмена установки рабочей книги в качестве дополнительной надстройки;
– ***WorkbookBeforeClose*** – возникает непосредственно перед закрытием рабочей книги;
– ***WorkbookBeforePrint*** – возникает непосредственно перед печатью рабочей книги;
– ***WorkbookBeforeSave*** – возникает непосредственно перед сохранением рабочей книги;
– ***WorkbookDeactivate*** – возникает, когда открытая рабочая книга теряет фокус;
– ***WorkbookNewSheet*** – возникает при добавлении нового рабочего листа;
– ***WorkbookOpen*** – открытие рабочей книги.

Чтобы обработать событие объекта ***Application***, необходимо создать новый *модуль класса* и объявить в нем переменную, используя ключевое слово ***WithEvents***:

```
Public WithEvents ПеременнаяПриложения As Application
```


Данная инструкция определяет, что описанная в ней переменная используется для обработки событий.

Затем требуется связать описанный в модуле класса объект с элементом ***Application***. Это можно сделать в любом модуле, записав в нем следующий код:

```
Dim ПеременнаяСобытия As New ИмяМодуляКласса
Sub InitializeAppEvents()
  Set ПеременнаяСобытия.ПеременнаяПриложения = Application
End Sub
```

После выполнения данной процедуры объект, описанный в модуле класса, указывает на объект Excel ***Application***, что позволяет использовать для него процедуры обработки событий, описанные в модуле класса, при возникновении этих событий.

Если необходимо включить или отключить обработку событий, то следует изменить в программе свойство ***EnableEvents*** объекта ***Application***. Процедуры обработки событий доступны, если данное свойство равно ***True***.

Общий синтаксис заголовка процедуры обработки события можно представить в следующем виде:

```
Private Sub ПеременнаяПриложения_Событие (СписокПараметров)
```

Параметры, передаваемые процедурам обработки событий, позволяют уточнить условия возникновения и обработки событий (например, могут задать ссылку на рабочую книгу, с которой связано событие). Например, для обработки событий создаем модуль класса с именем «EventClassModule». В этом модуле класса можно сделать следующие описания:

```
Public WithEvents AppObj As Application
Private Sub AppObj_NewWorkbook(ByVal Wb _
                    As Excel.Workbook)
     x = MsgBox("Создаем новую книгу " & Wb.Name, _
                                  vbOKOnly, "Excel")
End Sub
```

Это означает, что для объекта-приложения, с которым будет связана переменная **AppObj**, определена процедура обработки события ***NewWorkbook***, возникающего при создании новой книги. Присваиваемое книге имя выводится на экран с помощью функции ***MsgBox***.

Если в списке объектов в окне модуля класса выбрать строку «Class», то в списке процедур можно выбрать имена процедур «Initialize» и «Terminate». Редактор VBA автоматически вставит в модуль «заготовки» для этих процедур, если они еще не были записаны. Данные процедуры используются для инициализации (например, присваивания значений общим переменным) и уничтожения экземпляра класса.

Для создания процедуры обработки события следует выбрать в списке «Объект» имя переменной, описанной в модуле с ключевым словом ***WithEvents***, а в списке «Процедура» – нужное событие. Редактор VBA автоматически сгенерирует «заготовку» для процедуры обработки выбранного события (ее заголовок со стандартным именем и списком параметров и инструкцию, завершающую процедуру).

После создания модуля класса для обработчиков событий приложения можно в любом модуле «подключиться» к этим процедурам. Для этого нужно описать переменную

```
Dim EvObj As New EventClassModule
```

и записать код, который установит ссылку на приложение, для которого должны обрабатываться события:

```
Sub InitAppEvents()
   Set EvObj.AppObj = Application
End Sub
```

Более полную информацию о свойствах объекта ***Application***, методах и событиях, связанных с этим объектом можно посмотреть в справочной системе, открыв окно просмотра объектов.

Работая с приложениями Excel, в среде Excel пользователь имеет дело с рабочими книгами.

9.2.2. Объект Workbook

Объект ***Workbook*** представляет файл рабочей книги Excel. Он расположен на следующем уровне иерархии. С точки зрения разработчика он является контейнером, содержащим любое приложение VBA, созданное в среде Excel. Приложение содержится в отдельной рабочей книге, но может работать с несколькими рабочими книгами.

Любая установка свойства или вызов метода для объекта ***Workbook*** действуют на все приложение.

Рабочая книга может храниться в двух формах: в формате XLS (стандартная рабочая книга) и XLA (файл программы-надстройки). При создании файла XLA можно быть уверенным, что приложение является полностью откомпилированным. Кроме того, создание такого файла позволяет спрятать текст программы от ее пользователя.

В Excel можно работать с несколькими рабочими книгами, т.е. рабочие книги образуют семейство ***Workbooks***. Чаще всего используют *свойство* ***Count*** (число объектов ***Workbook*** в семействе, доступно только для чтения), *методы* ***Add*** (создание нового объекта семейства), ***Open*** (открытие существующей рабочей книги) и ***Close*** (закрытие рабочей книги).

Наиболее часто используемыми *свойствами* объекта ***Workbook*** являются следующие свойства:

- ***ActiveSheet*** – ссылка на активный лист заданной рабочей книги;
- ***Author*** – имя автора рабочей книги (может использоваться в программе установки или регистрации);
- ***HasRoutingSlip*** – управляет включением в рабочую книгу маршрутов рассылки по электронной почте (тип ***Boolean***);
- ***Name*** – имя рабочей книги (тип ***String***; свойство только для чтения, т.к. для изменения имени нужно сохранить файл с помощью метода ***SaveAs***)
- ***Path*** – путь к файлу рабочей книги (тип ***String***; только чтение);
- ***Saved*** – если имеет значение ***True***, то со времени последнего сохранения книги в ней не было сделано никаких изменений (тип ***Boolean***; доступно для чтения и записи).

Основные *методы* объекта ***Workbook***:

- ***Activate*** – активизирует первое окно, связанное с рабочей книгой и делает книгу активной;
- ***Close*** – закрывает рабочую книгу (с помощью аргументов можно указать, *сохранять ли* изменения перед закрытием и, если изменения сохраняются, *имя файла*, в котором

сохраняется рабочая книга, а также определить необходимость *рассылки* книги по электронной почте, если для нее задан маршрут рассылки);
- ***Protect*** – защищает рабочую книгу, т.е. блокирует возможность внесения изменений в нее (в качестве аргументов можно передать пароль и «режим» защиты: подлежит ли защите структура рабочих листов и окон);
- ***Route*** – пересылка рабочей книги другому пользователю, используя текущий маршрут рассылки;
- ***Save*** – сохраняет рабочую книгу на старом месте под тем же именем;
- ***SaveAs*** – сохраняет рабочую книгу на новом месте и/или под новым именем;
- ***SaveCopyAs*** – сохраняет рабочую книгу в новом файле, оставляя существующую копию в памяти с прежним именем.

Возможна обработка событий, возникающих при работе с рабочими книгами, при изменении объекта ***Workbook*** или при редактировании его рабочих листов. Для описания процедур обработки этих событий не требуется создание модуля класса. Чтобы вывести список событий рабочей книги и просмотреть или отредактировать процедуры их обработки, можно свернуть окно рабочей книги, щелкнуть правой кнопкой по заголовку свернутого окна и выбрать в контекстном меню команду **Исходный текст** (будет запущен редактор VBA). В окне редактора имя интересующего события можно выбрать из списка процедур объекта ***Workbook*** окна программы редактора VBA.

Чаще всего в приложениях обрабатываются следующие *события*:
- ***BeforeClose*** – возникает при закрытии рабочей книги, перед выводом запроса на сохранение изменений;
- ***BeforePrint*** – возникает непосредственно перед печатью рабочей книги или ее части;
- ***BeforeSave*** – возникает непосредственно перед сохранением рабочей книги;
- ***Deactivate*** – возникает, когда открытая рабочая книга теряет фокус;

- ***NewSheet*** – возникает при добавлении нового рабочего листа;
- ***Open*** – открытие рабочей книги;
- ***SheetActivate*** – возникает при активизации любого рабочего листа книги;
- ***SheetBeforeDoubleClick*** – возникает при двойном щелчке по любому рабочему листу книги, до выполнения стандартной операции, которая производится по умолчанию при двойном щелчке;
- ***SheetBeforeRightClick*** – возникает при щелчке правой кнопкой по любому рабочему листу книги, до выполнения стандартной операции, которая производится по умолчанию при нажатии правой кнопки мыши;
- ***SheetCalculate*** – возникает после пересчета значений на любом рабочем листе книги или после вывода измененных данных на диаграмме;
- ***SheetChange*** – возникает после изменения пользователем или ссылкой содержимого ячейки на любом рабочем листе;
- ***SheetDeactivate*** – возникает, когда активный рабочий лист теряет фокус;
- ***SheetSelectionChange*** – возникает при изменении выделенного диапазона ячеек.

Информация в рабочей книге располагается на листах.

9.2.3. Объект Worksheet

Рабочая книга является предком (родительским объектом) объекта ***Worksheet***. Рабочие листы – основа для конструирования форм, выполнения задач обработки данных и вывода их на экран. Рабочие листы могут использовать свыше 400 встроенных функций Excel. Табличная структура рабочего листа делает его прекрасным средством для хранения информации, небольших наборов данных.

Множество всех объектов ***Worksheet*** представляется семейством ***Worksheets***. Элементы этого семейства являются подмножеством семейства ***Sheets***, включающего не только рабочие листы, но и листы диаграмм, диалоговых окон Excel 5.0. Наиболее

часто используемым *свойством объекта* **Worksheets** является свойство **Count** (число объектов в семействе). *Метод* **Add** (создание рабочего листа) – это чаще всего используемый метод семейства.

Чаще всего используются следующие *свойства* объекта **Worksheet**:

- **Index** – индекс заданного объекта **Worksheet** в семействе объектов **Worksheets** (только чтение);
- **Name** – имя рабочего листа;
- **UsedRange** – возвращает объект **Range**, который ссылается на диапазон рабочего листа, содержащий данные;
- **Type** – тип рабочего листа;
- **Visible** – управляет выводом рабочего листа на экран.

Основные методы *объекта* **Worksheet**:

- **Activate** – активизирует заданный рабочий лист;
- **Calculate** – вызывает перевычисление всех формул рабочего листа;
- **Copy** – копирование рабочего листа;
- **Delete** – удаляет рабочий лист из книги;
- **Move** – перемещение рабочего листа;
- **Protect** – защищает рабочий лист.

Рабочий лист состоит из ячеек, для работы с которыми существуют соответствующие объекты.

9.2.4. Объект Range

Объект **Range** содержится в объекте **Worksheet** и используется для представления одной или более ячеек рабочего листа. Непосредственно из ячеек доступны свыше 400 встроенных функций Excel, из них можно вызывать функции VBA, ячейки можно связывать с другими ячейками того же листа, других рабочих листов и других рабочих книг.

Этот объект «попадает между» единичными объектами и объектами из семейств. При ссылке на ячейки диапазона можно указывать *адрес* единичной ячейки (ее *координаты*, например: **Range("A1")**), *адресный индекс* (например: **Range("A1:F15″)**) или *имя*, присвоенное диапазону (например, после выполнения присваивания **Range("A1").Name="Ячейка1″** можно обращаться к ячейке по имени: **Range("Ячейка1″)**).

Диапазон в программе *можно задать* следующим образом:

- используя ссылку на ячейку (например: **Range("D15")**);
- используя имя диапазона (например: **Range("Итоги")**);
- используя объект **ActiveCell**, указывающий на ячейку или объект **Range**, который имеет фокус при вводе данных с клавиатуры;
- используя объект **Selection** (в отличие от объекта **ActiveCell**, может указывать не только на одну ячейку или даже диапазон ячеек, но и на графические объекты и т.п.).

Основные *свойства* объекта **Range**:

- **Address** – расположение диапазона на рабочем листе (параметры определяют, возвращается ли внешняя ссылка, абсолютные или относительные координаты, стиль ссылок (нотация A1 или нотация R1C1));
- **Cells** – ссылка на ячейки по их положению относительно заданного объекта **Range**;
- **Column** – номер первого столбца в диапазоне;
- **CurrentRegion** – объект **Range**, включающий диапазон ячеек, границами которого являются пустые строки и столбцы (полезен, например, если диапазон выделяемых данных изменяется – нужно выделить весь массив данных на листе);
- **Count** – количество ячеек, представляемых объектом (только для чтения);
- **Dependents** – диапазон ячеек, содержащих все зависимости исходного диапазона ячеек, т.е. все ячейки, в формулах которых есть ссылки на какие-либо ячейки из исходного диапазона (объект **Range**; только чтение);
- **Name** – имя диапазона;
- **Row** – номер первой строки в диапазоне;
- **Value** – значение, которое содержит диапазон (если в диапазоне несколько ячеек, то значением свойства является массив, содержащий значения всех ячеек диапазона);

– ***Formula*** – представленная в виде текстовой строки формула (или массив строк, содержащих формулы), которая содержится в диапазоне ячеек (включая знак “=”);
– ***Text*** – представленное в виде текстовой строки форматированное значение (или массив строк, содержащих форматированные значения), содержащееся в диапазоне (только чтение).

Основные *методы* объекта ***Range***:

– ***AutoFit*** – задание автоматической настройки ширины столбца и высоты строки диапазона;
– ***Calculate*** – вызывает перевычисление всех формул диапазона;
– ***ClearContents*** – очищает все значения и формулы диапазона, но оставляет форматы;
– ***ClearComments*** – очищает комментарии;
– ***ClearFormats*** – удаляет формулы;
– ***ClearNotes*** – удаляет примечания;
– ***Copy*** – копирует значения из диапазона ячеек в другой диапазон или в буфер;
– ***Cut*** – вырезает значения из диапазона ячеек и помещает их в другой диапазон или в буфер;
– ***Count*** –число объектов (в диапазоне);
– ***PasteSpecial*** – специальная вставка из буфера обмена (реализация команды специальной вставки).

Объект ***Range*** можно использовать совместно с семействами ***Rows*** и ***Columns*** (это свойства рабочих листов или диапазонов, возвращающие объекты ***Range***, представляющие все строки и все столбцы рабочего листа или диапазона).

Номера первой строки и первого столбца являются свойствами диапазона. Чтобы вернуть номер последнего столбца в диапазоне, можно записать следующее выражение:

```
Диапазон.Columns(Диапазон.Columns.Count).Column
```

Аналогично можно получить номер последней строки.

9.2.5. Графические объекты

Объект ***Shape*** представляет любой графический объект, размещенный в верхнем слое рабочего листа или диаграммы. Все объекты ***Shape*** рабочего листа представлены семейством ***Shapes***, которое может содержать такие разные графические объекты, как растровые изображения и другие виды рисунков, прямоугольники, линии, объекты WordArt и даже элементы управления. За исключением элементов управления пользователь может создавать все объекты ***Shape***, используя Панель рисования.

Каждый тип объекта ***Shape*** имеет свой собственный уникальный набор свойств и методов, но все объекты имеют и набор общих свойств и методов. Этот набор, используемый всеми объектами, представлен ниже.

Свойства объекта ***Shape***:

– ***Left***, ***Top***, ***Width***, ***Height*** – определяют положение и размер фигуры на рабочем листе или диаграмме;
– ***Name*** – текстовая строка, используемая для идентификации объекта (может использоваться в качестве индекса в семействе); это свойство можно задать «вручную», вписав значение в поле «Имя» строки формул Excel;
– ***Type*** – константа, идентифицирующая тип объекта (например, ***msoLine***, ***msoPicture*** и т.п.);
– ***Fill*** – возвращает объект ***FillFormat***, содержащий свойства для задания цвета, типа штриховки и текстуры объекта;
– ***Line*** – возвращает объект ***LineFormat***, содержащий свойства для настройки цвета, толщины и типа граничных линий объекта;
– ***Shadow*** – возвращает объект ***ShadowFormat***, содержащий свойства для настройки тени объекта;
– ***TextFrame*** – возвращает объект, содержащий подобъекты для задания и форматирования текста, находящегося в объекте ***Shape***;
– ***Visible*** – определяет, будет ли объект виден пользователю;

– ***OnAction*** – служит для запоминания имени подпрограммы VBA, которая будет запущена, если щелкнуть на объекте мышью.

Методы объекта ***Shape***:

– ***Copy*** – копирует объект в буфер обмена (Clipboard);
– ***Cut*** – копирует объект в буфер обмена (Clipboard) и удаляет его с рабочего листа или диаграммы;
– ***Flip*** – располагает объект горизонтально или вертикально;
– ***Delete*** – удаляет объект;
– ***IncrementLeft*** – сдвигает объект по горизонтали;
– ***IncrementTop*** – сдвигает объект по вертикали;
– ***IncrementRotation*** – поворачивает объект.

Методы семейства **Shapes**:

– ***AddShape*** – добавляет объект в семейство;
– ***AddPicture*** – добавляет в семейство объект типа рисунка;
– ***AddLabel*** – добавляет в семейство объект типа надписи.

Объект ***ShapeRange*** представляет произвольную группу фигур (такая группа получается при множественном выделении), но включенные в него объекты могут быть не выделены, таким образом, объект ведет себя как семейство. Этот объект можно создать с помощью семейства ***Shapes***, вызвав метод ***Range*** и передав ему в качестве параметра массив имен объектов ***Shape***. Созданным семейством можно манипулировать как одним объектом. Все объекты группы можно обработать в цикле ***For Each***.

9.2.6. Элементы управления и обработка событий

Кроме свойств и методов, элементы управления имеют предопределенный набор «событий». *Событие* – это действие со стороны пользователя (щелчок мышью, например) или системы (событие таймера и т.п.). *Процедура обработки события* – это подпрограмма, реализующая реакцию на это действие. Таким образом, обработку события можно реализовать, написав процедуру VBA.

Элементы управления могут находиться в диалоговом окне. Их можно размещать и на рабочем листе.

Объекты, представляющие элементы управления, используются всеми приложениями MS Office. Назначение объектов, представляющих элементы управления, основные их свойства и связанные с ними события рассматривались выше, при обсуждении интерфейса пользователя. Там же было показано, как можно создать диалоговые окна пользователя и получить доступ к стандартным диалоговым окнам.

Элементы управления можно разместить прямо на рабочих листах Excel. Для этого используются панели инструментов «Элементы управления» и «Формы» (рис. 9.5).

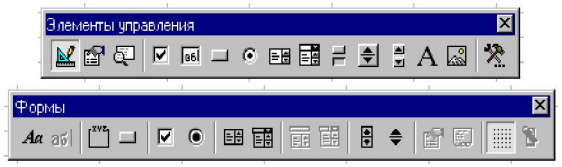


Рис. 9.5. **Панели, содержащие элементы управления**

Объекты, доступные через панель «Элементы управления» – это элементы ActiveX, имеющие несколько событий, обработку которых можно организовать через VBA; эти элементы можно использовать на рабочих листах и пользовательских формах. Элементы управления панели «Формы» – это стандартные элементы управления рабочего листа Excel; они имеют фактически только одно событие (***Click***), на которое может быть установлена реакция, если связать подпрограмму со свойством ***OnActive***. Эти элементы можно использовать на листах и диаграммах.

Для размещения элемента управления на рабочем листе его нужно выбрать на панели щелчком мыши (курсор мыши примет форму значка ‘+’) и «нарисовать» его с помощью мыши, как обычный графический объект.

Если элемент выбран с панели «Элементы управления», то все его свойства можно просмотреть и изменить, выполнив команду **Свойства** в контекстном меню или с помощью кнопки на панели. Для разработки кода процедур обработки событий, связанных с элементом управления, следует выполнить команду **Исходный текст** в контекстном меню объекта или щелчком по соответствующей кнопке на панели. При этом раскроется окно

редактора VBA, в котором можно выбрать из списка имя обрабатываемого события для данного объекта. При выборе события редактор автоматически будет создавать «заготовку» для процедуры обработки события, содержащую заголовок процедуры и завершающую ее инструкцию. После написания кода следует перейти снова в Excel.

При размещении элемента управления и определении его свойств в Excel включен режим конструктора (на панели «нажата» кнопка ). Для запуска процедур, связанных с событиями, определенными для элемента управления, следует выйти из режима конструктора, щелкнув по кнопке **Режим конструктора** на панели. После этого вставленный элемент управления начнет «работать», будут обрабатываться связанные с ним события.

Для изменения свойств созданных элементов управления нужно снова войти в режим конструктора.

Если элемент выбран на панели «Формы», его свойства также доступны через кнопку **Свойства элемента управления** на панели или команды **Изменить текст** и **Формат объекта** в контекстном меню. Установить процедуру обработки события можно с помощью команды **Назначить макрос** в контекстном меню или с помощью кнопки **Текст программы** на панели после выделения объекта (для выделения можно перейти в режим выбора объектов с помощью кнопки на панели рисования, вернуться в рабочий режим можно, «отжав» эту кнопку).

Элементы управления, размещенные на рабочем листе, можно связать с конкретными ячейками или диапазонами ячеек рабочего листа. Таким образом можно создать удобную пользовательскую форму для ввода и обработки информации.

Основой для создания форм Excel служит объект ***Worksheet***, т.е. форма строится на основе рабочего листа (в отличие от диалоговых окон, которые строятся на основе пользовательских форм, т.е. объекта ***UserForm***). Форма Excel является особым образом отформатированным рабочим листом. Поэтому чаще всего при создании формы используется объект ***Range*** и его подобъекты, а также объект ***Window***.

Свойства объектов ***Workbook*** и ***Worksheet*** рассматривались выше. Далее рассматриваются события, связанные с этими объектами.

Процедуры обработки событий объекта ***Worksheet*** находятся в модуле кода каждой рабочей книги. Этот объект поддерживает следующие события:

- ***Activate*** – происходит при активизации листа;
- ***BeforeDoubleClick*** – происходит после двойного щелчка мышью на листе, но перед реакцией Excel по умолчанию (эту реакцию можно отменить, установив аргумент ***Cancel*** в ***True***); аргумент ***Target*** указывает ячейку, на которой был произведен щелчок;
- ***BeforeRightClick*** – происходит после щелчка правой кнопкой мыши на листе, но перед реакцией Excel по умолчанию (эту реакцию можно отменить, установив аргумент ***Cancel*** в ***True***); аргумент ***Target*** указывает ячейку, на которой был произведен щелчок;
- ***Calculate*** – происходит после пересчета листа;
- ***Change*** – происходит при изменении данных на листе; аргумент ***Target*** указывает ячейку, в которой было произведено изменение;
- ***Deactivate*** – происходит, когда лист перестает быть активным;
- ***SelectionChange*** – происходит при изменении вычисления.

Процедуры обработки событий объекта ***Workbook*** находятся в модуле кода «This Workbook». Этот объект поддерживает следующие события:

- ***Activate*** – происходит при активизации рабочей книги;
- ***AddinInstall*** – происходит при установке рабочей книги как надстройки;
- ***AddinUninstall*** – происходит при отмене установки рабочей книги как надстройки;
- ***BeforeClose*** – происходит перед закрытием рабочей книги (для отмены закрытия можно установить аргумент ***Cancel*** в ***True***);
- ***BeforePrint*** – происходит перед печатью или отображением предварительного просмотра книги (для отмены печати можно установить аргумент ***Cancel*** в ***True***);

– ***BeforeSave*** – происходит перед сохранением рабочей книги (для отмены сохранения можно установить аргумент ***Cancel*** в ***True***);
– ***Deactivate*** – происходит, когда рабочая книга перестает быть активной;
– ***NewSheet*** – происходит при вставке нового листа в рабочую книгу (аргумент ***Sh*** является объектом, указывающим на новый лист);
– ***Open*** – происходит при открытии рабочей книги (используется вместо старого макроса Auto_open из предыдущих версий Excel);
– ***WindowActivate*** – происходит при активизации окна рабочей книги;
– ***WindowDeactivate*** – происходит при деактивизации окна рабочей книги;
– ***WindowResize*** – происходит при изменении размера окна рабочей книги (разворачивание, сворачивание, восстановление или изменение размеров вручную).

Порядок создания процедур обработки событий для различных объектов Excel (приложений, рабочих книг и листов) рассматривался выше.

9.3. Представление данных и вычисления в таблицах Excel

Электронные таблицы предназначены для представления данных в виде таблиц, реализации вычислений в соответствии с формулами и правилами, задаваемыми пользователем.

Excel позволяет вводить данные в ячейки таблицы как непосредственно, так и через строку формул (в ячейке таблицы данные отображаются в соответствии с форматом, заданным для этой ячейки пользователем, а в строке формул – в том виде, в каком данные были введены). Для автоматизации ввода данных в Excel используются команды заполнения: ячейки можно заполнить путем *копирования* ранее введенных в некоторый диапазон значений, *числовыми последовательностями* (арифметической и геометрической прогрессиями), *списками* (в Excel определены списки, представляющие названия дней недели, месяцев, кроме того, пользователь может задать свои списки), *последовательностями дат*. Заполнение может выполняться как с помощью мыши, так и с помощью команды **Заполнить** меню **Правка**. Более подробную информацию о способах заполнения можно получить в разделе «Ввод данных» справочной системы.

Для автоматизации вычислений в ячейки таблицы помещаются *формулы*.

Между данными, расположенными в различных таблицах, могут быть установлены *связи*. Это позволяет автоматически обновлять связанные данные во всех таблицах при изменении исходных данных, что позволяет осуществлять ввод данных однократно, избежать дублирования, поддерживать целостность (согласованность) данных в нескольких таблицах.

Таблица Excel может быть получена путем *консолидации* данных из нескольких таблиц (исходных диапазонов).

Данные в ячейках таблицы отображаются в соответствии с заданным пользователем *форматом*. Для определения формата используются команды меню **Формат**.

9.3.1. Ссылки на ячейки рабочих листов Excel

Координаты (адрес *ячейки*, *ссылка* на нее) однозначно определяют ячейку или группу ячеек листа. Кроме того, можно ссылаться на ячейки, находящиеся на других листах книги или в другой книге, или на данные другого приложения. Ссылки на ячейки других книг называются *внешними* ссылками. Ссылки на данные других приложений называются *удаленными* ссылками.

Координаты текущей (активной) ячейки или ее имя (если имя определено) отображаются в поле со списком «Имя» в окне Excel. Введя координаты в это поле или выбрав имя из списка определенных имен, можно быстро перейти к нужной ячейке.

Ссылки на ячейки используются в формулах для выбора значений ячеек, используемых при вычислениях.

По умолчанию в Microsoft Excel используются ссылки вида “A1”, в которых столбцы обозначаются буквами от A до IV (256 столбцов максимально), а строки числами – от 1 до 65536). Чтобы указать ссылку на ячейку, вводится буква заголовка столбца, а затем номер строки. Например, D50 является ссылкой на ячей-

ку, расположенную на пересечении столбца D с 50-й строкой.

Можно воспользоваться стилем ссылок, в котором и столбцы, и строки листа пронумерованы. Этот стиль, называемый “R1C1”, наиболее полезен при вычислении позиции строки и столбца в макросах, а также при отображении относительных ссылок. В стиле “R1C1”, после буквы ‘R’ указывается номер строки ячейки, после буквы ‘C’ – номер столбца.

Стиль ссылок можно изменить с помощью команды **Параметры** меню **Сервис**.

Excel может использовать *относительные* и *абсолютные* координаты ячеек.

Абсолютные координаты не изменяются, когда ячейка, содержащая формулу (в частности, функцию), копируется в другое место. Для указания абсолютных координат ячеек перед координатой (номером строки или названием столбца) следует поместить знак доллара (символ $), например: $B$5 (абсолютные координаты ячейки B5). В стиле “R1C1” указывается место размещения ячейки, где после буквы ‘R’ ставится номер строки ячейки, а после буквы ‘C’ – номер столбца. Например, абсолютная ссылка “R1C1” эквивалентна абсолютной ссылке $A$1 для формата “A1”.

Однако в большинстве случаев удобно, чтобы при копировании содержимого ячейки, содержащей формулу со ссылками на другие ячейки, их координаты изменялись бы (например, значение ячейки – сумма значений предшествующих ей ячеек той же строки; если значения всех ячеек этого столбца вычисляются по той же формуле, то при ее копировании координаты ячеек, содержащих слагаемые, должны изменяться, настраиваясь на новый номер строки). При копировании формулы в другое место относительные координаты изменяются. При определении относительных координат знак доллара добавлять не нужно. Если ячейка A1 является текущей, то относительная ссылка R[1]C[1] в стиле “R1C1” указывает на ячейку, расположенную на одну строку ниже и на один столбец правее, то есть на ячейку B2.

Если при создании формулы используется диапазон ячеек, то он вводится в формулу в такой последовательности: координаты первой ячейки диапазона, двоеточие и координаты последней ячейки диапазона (например, B1:B5).

Если в формуле используются значения ячеек других рабочих листов, то соответствующие операнды должны иметь следующий формат: имя рабочего листа, восклицательный знак, координаты ячейки (например, Лист1!B5).

Для ссылки на ячейки другой рабочей книги используются «трехмерные» ссылки вида '[Книга.xls]Лист'!Диапазон.

Чтобы избежать использования координат, сделать ссылки на ячейки более наглядными, ячейке или диапазону ячеек можно присвоить имя, которое затем можно будет использовать вместо указания координат ячеек.

Для определения имени ячейки или диапазона используется команда **Имя ▸ Присвоить** меню **Вставка**. После создания именованного диапазона вместо координат его ячеек (например, при воде формул или выборе диапазона) можно использовать введенное имя.

Координаты ячеек (номера строки и столбца) являются свойствами объекта ***Range***.

Свойством ***Range*** обладают объекты ***Application***, ***Range*** и ***Worksheet***. Значением этого свойства является объект ***Range***. Для обращения к этому свойству необходимо записать выражение вида

Выражение.*Range*(Cell1, Cell2)

или

Выражение.*Range*(Cell)

где **Cell** – имя диапазона или ссылка, которая должна быть задана в A1-стиле (задается как текстовая строка), или объект ***Range***, который может содержать единственную ячейку, целый столбец или строку; **Cell1** и **Cell2** определяют ячнйки в левом верхнем и правом нижнем углах диапазона. Например, операторы

Worksheets*(1).*Range*("A1:B3").*Select

и

Worksheets*(1).*Range*("A1", "B3").*Select

позволяют выделить один и тот же диапазон рабочего листа. Если диапазону “A1:B3” присвоено имя (например, “Range_name”), оператор выделения диапазона можно записать так:

***Worksheets*(1).*Range*("Range_name")**

Если **Выражение** не задано (т.е. для диапазона не указан квалифицирующий объект), то по умолчанию используется объект ***ActiveSheet***, т.е. в качестве значения возвращается диапазон активного рабочего листа.

Свойство ***Range*** зависит от объекта ***Range***. Например, если выделена ячейка **C3**, то результатом вычисления выражения

```
Selection.Range("B1")
```

станет ячейка **D3** (используется относительная ссылка, результат вычисляется по отношению к текущему выделению), а выражение

```
ActiveSheet.Range("B1")
```

всегда вернет значение **B1**.

При реализации вычислений в Excel в различных контекстах бывает удобнее пользоваться различными способами ссылок на ячейки.

Свойство ***Cells*** (***Cells***(**RowNum**, **ColumnNum**)) используется, когда нужно указать единственную ячейку в строке и столбце с заданными номерами (индексами **RowNum** и **ColumnNum**). Например, следующий оператор устанавливает для ячейки **A1** первого рабочего листа значение 100:

```
Worksheets(1).Cells(1, 1).Value = 100
```

В некоторых случаях данное свойство удобнее использовать, чем свойство ***Range***, например, если для указания строки и/или столбца используются целочисленные переменные, значения которых пересчитываются в программе. При использовании же свойства ***Range*** необходимо выполнять преобразование целочисленных переменных к строковому типу для указания ссылки на ячейку.

Ссылка, возвращаемая как результат при вычислении

```
Выражение.Cells(Row, Column)
```

где **Выражение** возвращает объект ***Range***, определяет ячейку по отношению к левому верхнему углу полученного диапазона. Например, оператор

```
Range("C5:C10").Cells(1, 1).Formula = "=СУММ(C6:C10)"
```

заносит в первую ячейку указанного диапазона (**C5**) активного рабочего листа формулу для вычисления суммы значений, записанных в диапазоне "**C6:C10**".

Свойство ***Offset*** (***Offset***(**RowOffs**, **ColumnOffs**)) используется, когда строка и столбец нужной ячейки определяются с помощью смещений (**RowOffs** и **ColumnOffs**)) по отношению к другой ячейке (диапазону). Например, после выполнения операторов

```
Application.Goto Reference:="R1C1"
Selection.Offset(3, 1).Range("A1").Select
```

выделенной окажется ячейка **B4** (происходит смещение вниз на 3 строки и на один столбец вправо по отношению к ячейке, находящейся в левом верхнем углу выделенного диапазона).

Метод ***Union*** (***Union***(**Range1**, **Range2**, ...)) используется для получения диапазонов, которые могут состоять из двух и более прямоугольных блоков ячеек (прямоугольных диапазонов **Range1**, **Range2**, ...). В приведенном ниже примере создается объект, являющийся объединением двух диапазонов – "**A1:B2**" и "**C3:D4**", после чего эти диапазоны выделяются.

```
Dim R1 As Range, R2 As Range, myMultiAreaRange As Range
Worksheets("sheet1").Activate
Set R1 = Range("A1:B2")
Set R2 = Range("C3:D4")
Set myMultiAreaRange = Union(R1, R2)
myMultiAreaRange.Select
```

Для выделения, которое состоит из более чем одной области, можно воспользоваться свойством ***Areas***, которое позволяет разбить выделение на отдельные диапазоны и возвращает соответствующие объекты как семейство. Количество объектов в семействе можно получить с помощью свойства ***Count***:

```
numberOfSelectedAreas = Selection.Areas.Count
```

Полученные с помощью ссылок объекты, их свойства и методы можно использовать в вычислениях с помощью формул, вводимых в ячейки рабочего листа Excel.

9.3.2. Работа с формулами

В ячейки рабочего листа можно ввести не только значения, но и *формулы* для вычисления значений. Формулы используются для расчетов значений ячеек, зависящих от значений, хранящихся в других ячейках рабочего листа. Формулы позволяют выполнять обычные операции над константами и значениями ячеек.

Операнды операций (аргументы) могут задаваться как константы (значения), координаты ячеек, из которых нужно выбирать значения для вычислений, или диапазоны (они могут быть использованы как аргументы некоторых функций в формулах).

Если при создании формулы в качестве операнда используется одна ячейка текущего листа, то в формулу войдут ее *координаты* (адрес *ячейки*, *ссылка* на нее).

В формулах можно использовать знаки различных операций (в табл. 9.2 операции приведены в порядке убывания их приоритетов при вычислении).

Ввод формулы начинается с ввода знака равенства (символа ‘=’), за которым вводятся операнды и операции.

По умолчанию Excel отображает на экране не формулы, а результаты вычислений по ним. Саму формулу можно увидеть в строке формул, поместив на соответствующую ячейку рамку выделения. Однако это неудобно, так как нужно помнить, в каких ячейках введены формулы, или перемещать рамку по всем ячейкам. Поэтому иногда нужно увидеть все формулы рабочего листа. Для этого нужно установить флажок **Формулы** в группе **Параметры окна** на вкладке **Вид** диалогового окна, которое открывается командой **Параметры** меню **Сервис**.

***Таблица 9.2*. Символы операций**

Символ	Операция
–	Унарный минус (отрицание)
%	Процент (вычисление процента)
^	Возведение в степень
* и /	Умножение и деление
+ и –	Сложение и вычитание
&	Конкатенация (объединение текстовых строк)
	Операции сравнения:
>	Больше
<	Меньше
=	Равно
<>	Не равно
<=	Меньше или равно
>=	Больше или равно

По умолчанию Excel заново пересчитывает значения по всем формулам рабочего листа каждый раз, когда вносятся изменения в ячейки, указанные в формуле. При больших размерах таблиц это может снизить производительность. Поэтому пользователь может поменять режим пересчета значений по формулам. Для этого нужно выполнить команду **Параметры** меню **Сервис**, выбрать вкладку «Вычисления» открывшегося диалогового окна и установить режим пересчета «вручную». Если установлен этот переключатель, то можно заставить Excel пересчитать все значения в любой момент, когда это необходимо, нажав клавишу *F9*.

При открытии и распечатке рабочих листов все значения пересчитываются автоматически независимо от установленного режима.

В формуле при ее вводе может быть допущена ошибка (типичные ошибки: деление на нуль, использование координат пустых ячеек, пропуск разделителей между операндами (аргументами), использование неверных координат ячеек).

Если при попытке вычисления по формуле произошла ошибка, то вместо значения (результата вычисления по формуле) в ячейке выводится сообщение об ошибке (строка, начинающаяся символом **#**, например: **#ДЕЛ/0!** – сообщение о попытке выполнить деление на 0; **#Н/Д!** – отсутствуют данные, необходимые для расчетов по формуле; **#ИМЯ?** – в формуле используется ссылка на несуществующее имя; **#ЧИСЛО!** – в формуле используется недопустимый числовой аргумент; **#ССЫЛКА!** – неверно указаны координаты ячейки; **#ЗНАЧ!** – неверный тип данных для вычисления).

Если же в ячейке вместо вычисленного значения показывается строка **###**, это означает, что значение ячейки не может быть отображено, так как не вмещается в отведенные позиции в заданном формате. В этом случае можно расширить ячейку или уменьшить размер шрифта.

Формулу можно определить программно, задав соответствующее свойство объекта (ячейки). Например:

```
ActiveCell.FormulaR1C1 = "=5+R[2]C"
```

или

```
ActiveSheet.Cells(2, 1).Formula = "=sum(b1:b5)"
```

Это позволяет задавать сложные алгоритмы вычислений значений ячеек, переопределяя формулы в приложениях.

9.3.3. Использование функций

Для вычисления значений, вводимых в ячейки, можно использовать *функции*. Функция позволяет выполнить соответствующие ей действия (по заданному алгоритму, формуле для вычислений) над группой указанных в качестве ее *аргументов* значений. Например, широко используется функция для вычисления суммы (**СУММ(аргументы)**).

Использование функций облегчает ввод данных, позволяет автоматизировать сложные расчеты.

Значение каждой функции определяется именем функции и аргументами: имя функции указывает ее назначение, определяет алгоритм вычисления; аргументы задают исходные данные для вычислений.

В качестве аргументов функций могут быть указаны обычные значения (константы) или координаты ячеек таблицы, координаты диапазонов или поименованные диапазоны. Координаты могут быть как абсолютными, так и относительными.

Если функция вычисляется для значений нескольких аргументов, то они перечисляются через разделитель – точку с запятой (;) (например: выражение **=СРЗНАЧ(10; 11; 12; 13; 14)** вычисляет среднее значение из всех перечисленных в скобках значений) или указывается диапазон ячеек, значения которых должны рассматриваться как аргументы функции (выражение **=МИН(B1:B15)** дает минимальное значение из всех значений, содержащихся в заданном диапазоне ячеек).

Функции вводятся так же, как и значения: их можно ввести непосредственно в ячейке или в строке формул. Однако для облегчения ввода можно воспользоваться другими средствами (вызвать Мастер функций, использовать команду автосуммирования).

Вызовы функций в формуле могут быть вложенными (то есть в качестве аргумента какой-либо функции может быть использовано значение другой функции соответствующего типа).

9.3.4. Создание пользовательских функций рабочего листа

Возможности Excel можно расширить с помощью VBA, используя функции, разработанные пользователем.

Функции рабочего листа, определенные разработчиком, – это процедуры ***Function***, включенные в модуль проекта. Эти функции можно использовать в формулах, хранящихся в ячейках рабочих листов, точно так же, как и стандартные функции Excel. В частности, если при вычислениях используются сложные формулы, их можно реализовать в виде функций, записывая в ячейках вместо этих формул вызовы соответствующих функций с параметрами. Чтобы функции были доступны в рамках проекта, их следует описать с ключевым словом ***Public***. Функции, описанные в одной рабочей книге, можно использовать в других рабочих книгах, установив ссылки на них.

Ниже приведен пример использования пользовательской функции рабочего листа для вычисления надбавки в зависимости от разряда работника.

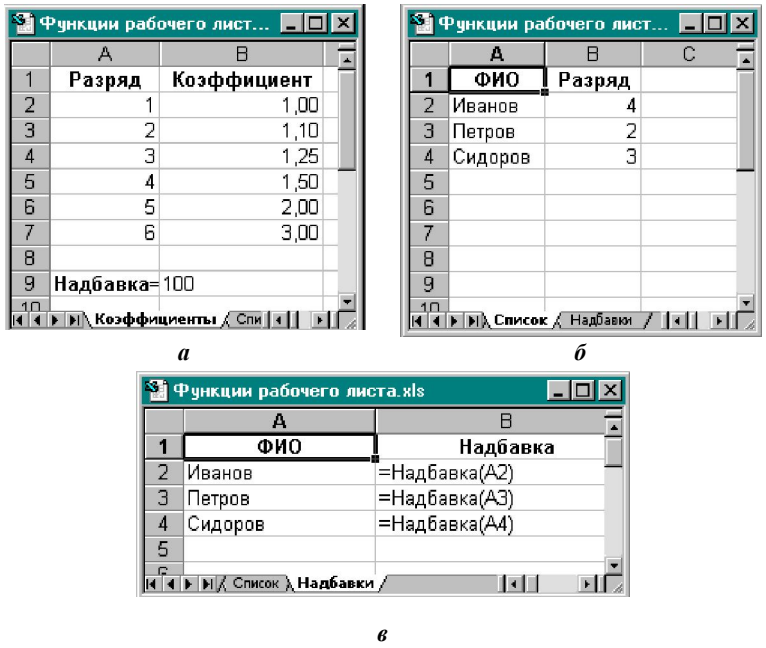
Для демонстрации возможности использования пользовательской функции рабочего листа создана рабочая книга, включающая три рабочих листа: «Список» (рис. 9.6, *а*), «Коэффициенты» (рис. 9.6, *б*) и «Надбавки» (рис. 9.6, *в*).

Надбавка вычисляется для каждого работника индивидуально. В качестве параметра в функцию передается ссылка на ячейку, в которой содержится фамилия работника. Порядок следования фамилий на рабочих листах «Список» и «Надбавки» одинаков. Коэффициенты, назначенные каждому разряду, на рабочем листе «Коэффициенты» упорядочены по возрастанию разрядов. Размер базовой надбавки всегда выбирается из одной и той же ячейки, расположенной на рабочем листе «Коэффициенты».

Для создания функции в проект добавлен модуль, в который затем вставлена функция:

```
Public Function Надбавка(ByVal CurrentCell As Range) _
                                    As Single
    Dim WR As Byte, _
        RN As Byte, _
        F As Single
```

```
    RN = CurrentCell.Row
    WR = Worksheets("Список").Cells(RN, 2).Value
    F = Worksheets("Коэффициенты").Cells(WR + 1, 2).Value
    Надбавка = F * Worksheets("Коэффициенты"). _
                                Cells(9, 2).Value
End Function
```



а *б*

в

Рис. 9.6. **Использование функций рабочего листа**

В результате вычислений надбавок в соответствии с описанной функцией получаются следующие результаты:

ФИО	Надбавка
Иванов	150,00
Петров	110,00
Сидоров	125,00

В вычислениях могут быть использованы значения не только из данной рабочей книги, но и из внешних источников. Доступ к внешним данным рассматривается ниже.

9.3.5. Массивы Excel

Массив Excel – это объект, используемый для вычисления нескольких значений в результате вычисления одной формулы или для работы с набором элементов, расположенных в различных ячейках и сгруппированных по строкам или столбцам. Существует два типа массивов: *диапазоны массива* и *диапазоны констант*. иапазоном массива называется непрерывный диапазон ячеек, использующих общую формулу. Диапазон констант представляет собой набор констант, используемых в качестве аргументов функций.

Формула массива может выполнить несколько вычислений, а затем вернуть одно значение или группу значений. Формула массива воздействует на несколько наборов значений, называемых аргументами массива. Каждый аргумент массива должен иметь соответствующий номер строки и столбца.

Рассмотрим пример: требуется определить число полученных зачетов и средние оценки как среди всех студентов, сдавших экзамен, так и только среди тех, кто получил зачет. Рабочий лист, содержащий исходные данные и результаты вычислений, показан на рис. 9.7.

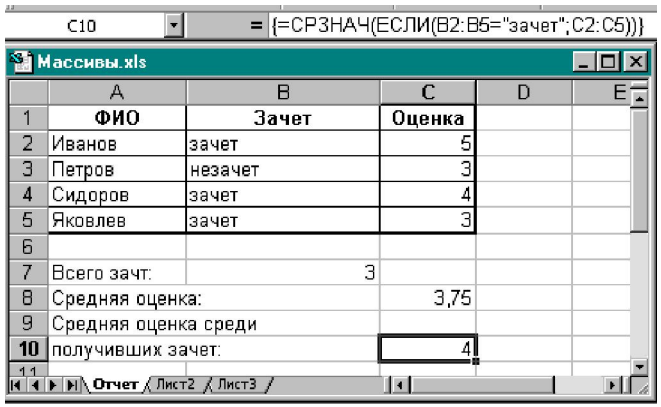


Рис. 9.7. **Использование формулы массива**

Общее количество зачетов определяется с помощью функции Excel (формула **=СЧЁТЕСЛИ(B2:B5;"=зачет")** введена в ячейку **B7**). Средняя оценка вычисляется с помощью формулы

=СРЗНАЧ(C2:C5), введенной в ячейку **C8**. А вот для вычисления средней оценки только для студентов, сдавших зечет, необходимо использовать формулу массива.

Формула массива создается так же, как и простая формула. Выделяется ячейка или группа ячеек, в которых необходимо создать формулу (в данном примере это **C10**), вводится формула (**=СРЗНАЧ(ЕСЛИ(B2:B5="зачет";C2:C5))**), показанная на рис. 9.7 в строке формул, а затем нажимаются клавиши *Ctrl+Shift+Enter* (формула вводится без фигурных скобок). После нажатия комбинации клавиш введенная формула в строке формул заключается в фигурные скобки, принимая вид

{=СРЗНАЧ(ЕСЛИ(B2:B5="зачет";C2:C5))}

что является признаком того, что формула интерпретируется Excel как формула массива. Если фигурные скобки ввести вручную, формула будет распознаваться Excel как символьная строка и вычисления не будут выполняться.

Введенная формула вычисляет среднее значение только тех ячеек, принадлежащих диапазону **C2:C5**, которым в соответствующих строках столбца B поставлено в соответствие значение **"зачет"**. Функция ЕСЛИ находит ячейки в диапазоне **B2:B5**, содержащие строку **"зачет"**, и возвращает значения, соответствующие этой строке в диапазоне **C2:C5**, передавая их функции **СРЗНАЧ**.

Для вычисления нескольких значений в формуле массива, необходимо ввести массив аргументов для вычисления по формуле в диапазон ячеек, имеющих соответствующее число строк или столбцов. Чтобы отобразить все вычисленные значения, необходимо ввести формулу в ячейки, в которые должны быть помещены результаты (рис. 9.8).

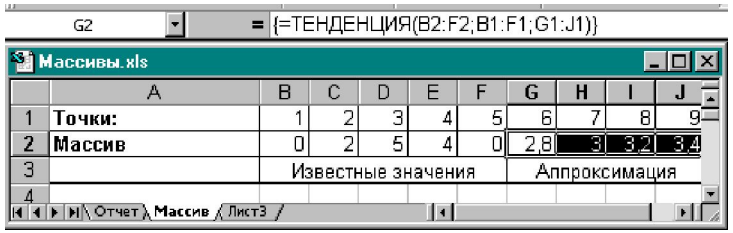


***Рис. 9.8*. Массив, вычисленный по формуле массива**

В приведенном на рис. 9.8 примере формула введена в ячейки диапазона "**G2:J2**".

Для генерации формулы массива в некоторых случаях можно воспользоваться Мастером частичных сумм, который позволяет пользователю создать формулу для вычисления суммы значений, находящихся в заданном столбце, причем суммироваться должны только значения, находящиеся в строках, ячейки которых удовлетворяют определяемым пользователем условиям. Это средство применимо к спискам (базам данных) Excel, так как каждый столбец таблицы должен иметь заголовок. Активизировать Мастер частичных сумм можно с помощью команды **Мастер ‣ Частичная сумма…**. Все параметры, определяющие условия суммирования задаются в диалоге. Это средство Excel рассматрвается ниже.

В формулу массива можно включать константы так же, как это делается в простой формуле, но массив констант должен вводиться в определенном формате. Более подробную информацию можно получить в справочной системе.

9.3.6. Определение связей между таблицами

Связи позволяют использовать на листе одной рабочей книги данные других листов и даже рабочих книг. Можно установить связь с одной ячейкой, диапазоном ячеек, диапазоном рабочих листов.

Рабочая книга, содержащая исходные данные, называется *исходной книгой* или *книгой-источником*. Книга, получающая данные, – это *книга-получатель*, или *целевая рабочая книга*.

Благодаря возможности связывать данные друг с другом, можно создавать небольшие рабочие книги с небольшими листами, соответствующие отельным задачам. Затем эти книги можно связать друг с другом для создания больших информационных систем, в которых информация не будет дублироваться и всегда будет поддерживаться в соглавосанном состоянии, что позволяет повысить гибкость в организации работы (связанные данные могут располагаться на различных компьютерах в сети), облегчить внесение изменений (изменения нужно вносить только в исходные данные).

Чтобы *связать ячейки или диапазоны ячеек* нужно выделить исходный диапазон ячеек, открыв исходную книгу, и выполнить команду копирования. Затем следует переключиться на целевой рабочий лист и выделить ячейку, которая должна быть расположена в левом верхнем углу диапазона ячеек, для которых создается связь. Затем нужно выполнить команду **Специальная вставка** меню **Правка**. В открывшемся диалоговом окне следует установить значения нужных переключателей (например, установить переключатель «Вставить» в положение «Все», а переключатель «Операция» – в положение «Нет»). Для завершения операции нажимается кнопка **Вставить связь**. В результате в ячейках будут отображены значения ячеек исходного диапазона, а содержимым этих ячеек (его можно увидеть в строке формул) будут соответствующие ссылки (рис. 9.9).

Рис. 9.9. **Рабочий лист с данными, полученными по связям**

Программным способом эти действия выполняются с помощью последовательности операторов (целевая книга к моменту, когда начинают выполняться операторы, должна быть открыта):

```
Workbooks.Open _
    FileName:="D:\Примеры\Исходные данные для ссылок.xls"
Sheets("Данные для копирования").Range("Источник").Select
Selection.Copy
Windows("Создание ссылок.xls").Activate
Sheets("Лист1").Select
Range("A1").Select
ActiveSheet.Paste Link:=True
```

При изменении исходных данных автоматически будет происходить обновление информации в связанных диапазонах.

Установить связь можно с помощью непосредственного указания: следует выделить целевую ячейку, в которую будет помещена связь, и обычным образом ввести формулу, в которой

будет задана нужная ссылка. Ввод завершается нажатием клавиши *Enter*. Таким образом, формулу, содержащую ссылку (связь), можно набрать вручную.

При *сохранении* рабочих книг, содержащих внешние ссылки, следует сначала сохранять исходные книги, а затем целевые (это обеспечит правильное запоминание пути к источнику). При изменении книги-источника лучше держать открытой книгу-получателя (при этом в целевой книге будут зафиксированы изменения).

Изменение информации в связанных книгах лучше выполнять с самых нижних уровней иерархии (от источников) и обновлять связи в порядке от книги-источника самого низшего уровня к целевой книге самого высокого уровня иерархии связей.

Если при *открытии целевой рабочей книги* с установленными связями исходная рабочая книга открыта, то связи обновляются автоматически. Если же исходная книга закрыта, то пользователь должен будет ответить на вопрос, хочет ли он работать с данными, оставшимися после последнего сохранения книги, или их следует обновить, выбрав новые значения из исходной рабочей книги.

Если связи при открытии *всегда должны обновляться*, то следует отключить сохранение копий внешних данных вместе с рабочей книгой. Для этого нужно выполнить команду **Параметры** меню **Сервис** и на вкладке «Вычисления» диалогового окна снять флажок «Сохранять значения внешних связей».

Если целевая рабочая книга уже открыта, то для *открытия исходных книг* можно выполнить команду **Связи** в разделе меню **Правка**. В открывшемся диалоговом окне приведен список всех источников, в нем можно выделить нужные файлы и щелкнуть кнопку **Открыть**.

В этом же диалоговом окне можно выделить источники, связь с которыми нужно *обновить* (заменить на другой файл, например). Обновление ссылок необходимо, если изменилось имя или расположение исходной рабочей книги. Операция обновления связей запускается щелчком по кнопке **Изменить**. В открывшемся диалоговом окне, в его верхней части, выделяется обновляемая связь, а в полях ввода в нижней части окна указывается новый, предназначенный для замены источник.

Внешние ссылки, связывающие ячейки или диапазоны ячеек, могут быть «заморожены» – внешняя ссылка удаляется, но сохраняются значения полей, *ссылка заменяется на значение*. Для выполнения этой операции следует указать нужную ячейку, нажить клавиши *F2* и *F9* (и кнопку **Вычислить**).

Существует еще один вид связывания – *связывание изображений ячеек*.

Более подробную информацию об используемых командах можно получить в справочной системе. Соответствующий код можно сгенерировать с помощью средств макрогенерации.

9.3.7. Консолидация данных

При *консолидации* происходит обобщение однородных данных, расположенных в различных таблицах, которые могут находиться на разных рабочих листах и даже в разных рабочих книгах. При этом могут вычисляться суммы, статистические величины и т.п.. Например, таким способом можно обрабатывать данные, поступающие в виде электронных таблиц с одинаковой структурой от различных филиалов, подводя общие результаты.

Существуют разные способы консолидации: консолидация трехмерными формулами, консолидация рабочих листов по физическому расположению, консолидация рабочих листов по заголовкам строк и столбцов.

Консолидацию с трехмерными формулами можно использовать, если есть рабочие книги, в каждой из которых есть рабочие листы с одинаковыми именами и консолидируемые данные размещаются в одних и тех же диапазонах в них, а результат консолидации размещается в одной ячейке и вычисляется в соответствии с известной формулой на основе данных из исходных диапазонов.

Рассмотрим пример. Есть две рабочие книги – «Расходы-1999.xls» и «Расходы-2000.xls», в каждой из которых на отдельном рабочем листе приведены сведения о расходах филиалов некоторой компании по месяцам (данные за каждый квартал находятся на отдельных листах, данные о каждом филиале расположены в одних и тех же строках, данные по месяцам – в отдельных столбцах, рис. рис. 9.10).

E7 = =СУММ(E2:E5)

Расходы-1999.xls

	A	B	C	D	E
1	Филиал	Январь	Февраль	Март	Всего
2	Северный	103 985,00р.	50 387,00р.	59 840,00р.	214 212,00р.
3	Южный	90 432,00р.	70 497,00р.	86 549,00р.	247 478,00р.
4	Восточный	150 367,00р.	123 004,00р.	34 892,00р.	308 263,00р.
5	Западный	80 476,00р.	49 203,00р.	104 993,00р.	234 672,00р.
6					
7	Итого:	425 260,00р.	293 091,00р.	286 274,00р.	1 004 625,00р.
8					

Квартал1 / Квартал2 / Квартал3 / Квартал4 /

Рис. 9.10. **Отчет о расходах за квартал**

Нужно создать рабочую книгу «Отчет.xls», в которой будут просуммированы расходы всех филиалов за каждый год. Для этого должны использоваться формулы суммирования с трехмерной ссылкой (Книга-Листы-Диапазон), которая для подсчетов расходов за 1999 г. будет выглядеть следующим образом:

=СУММ('[Расходы-1999.xls]Квартал1:Квартал4'!B2:D2)

В этой формуле производится суммирование всех данных за 1999 г. по одному из филиалов: трехмерная ссылка указывает исходную книгу и диапазон суммируемых ячеек (данных по месяцам) по всем четырем рабочим листам книги. Эта формула должна быть введена в ячейку, представляющую расходы первого филиала за 1999 год (B2), с клавиатуры или с помощью мыши. Введенную формулу затем можно скопировать в ячейки диапазона B3:B5. Аналогично вычисляются итоговые данные за 2000 г. (рис. 9.11).

B2 = =СУММ('[Расходы-1999.xls]Квартал1:Квартал4'!B2:D2)

Отчет.xls

	A	B	C	D	E
1	Филиал	Всего за 1999	Всего за 2000		
2	Северный	856848	1297756		
3	Южный	989912	1618096		
4	Восточный	1233052	1864636		
5	Западный	938688	1555472		
6					
7	Итого:	4018500	6335960		

Итоги /

Рис. 9.11. **Рабочий лист с консолидированными данными**

Вместо «ручного» ввода формулы в некоторых случаях можно выполнить консолидацию с помощью соответствующей команды меню **Данные**. Для этого в общем случае нужно:

- В целевой книге указать *диапазон назначения*, где консолидируются данные.
- Выполнить команду **Консолидация** в меню **Данные**.
- Последовательно указать исходные диапазоны данных, которые должны быть консолидированы (в строке ввода (в поле «Ссылка») диалогового окна консолидации формируется ссылка, которая добавляется в список диапазонов с помощью кнопки **Добавить**; можно задать до 255 диапазонов; «лишние» диапазоны можно удалить).
- Указать способ консолидации – по расположению в диапазоне или согласно заголовкам строк и столбцов. При консолидации по расположению нужно снять флажки «Подписи верхней строки» и «Значения левого столбца». Они устанавливаются, если соответствие при консолидации устанавливается по заголовкам.
- Задать режим формирования значений в диапазоне назначения: фиксированные значения, которые не будут изменяться в дальнейшем, или связанные величины, обновляющиеся при изменениях в исходных данных.
- Выбрать тип консолидации (функцию, которая должна выполняться над данными при их консолидации).

Заполнение целевого диапазона при выполнении консолидации будет происходить в зависимости от того, что было выделено в качестве диапазона назначения:

- Ячейка – заполняются все ячейки, необходимые для всех консолидируемых категорий исходных данных.
- Диапазон ячеек – консолидируется ровно столько категорий, сколько поместится в выделенном диапазоне.
- Строка ячеек – заполняются ячейки вниз от выделения по всей ширине выделенной области.
- Столбец ячеек – заполняются ячейки вправо от выделения по всей высоте выделенного диапазона.

Консолидация по расположению ячеек используется, если данные одного типа на всех листах расположены в одних и тех же позициях относительно исходных диапазонов, т.е. исходные диапазоны должны иметь абсолютно одинаковую структуру! *Данные в диапазоне назначения будут расположены так же, как и в исходных диапазонах* (т.е., например, будут суммироваться значения, расположенные в одних и тех же ячейках перечисленных исходных таблиц, а результаты будут помещаться в таблицу с той же структурой на те же места, где в исходных таблицах находятся слагаемые).

При консолидации по физическому расположению заголовки в диапазон назначения и исходные диапазоны для консолидации не включаются, так как Excel воспримет их как данные.

Примером консолидации такого типа может служить получение итоговых данных за год по всем филиалам путем суммирования итоговых данных за каждый квартал.

В качестве диапазона назначения на рабочем листе «Итог за год» нужно выделить диапазон B2:B5, а исходными диапазонами для консолидации должны стать диапазоны E2:E5 на рабочих листах, содержащих квартальные отчеты (в этих диапазонах находятся суммарные расходы каждого филиала за соответствующий квартал). Заголовки строк и столбцов нельзя включать в выделяемые диапазоны. Связи с исходными данными в этом случае также не установлены (рис. 9.12).

Рис. 9.12. **Консолидация по физическому расположению**

Ту же операцию можно было бы выполнить, используя копирование диапазона и специальную вставку (повторяя ее многократно, по очереди, для каждого диапазона).

Если при консолидации данных исходные диапазоны могут иметь различную структуру (в данных используются дополнительные поля, различные для консолидируемых таблиц или необходимо выборочно консолидировать лишь часть данных из исходных диапазонов и т.п.), то консолидацию по расположению использовать нельзя. В этом случае применим метод консолидации *по заголовкам строк и столбцов*.

Существует два способа такой консолидации.

Первый состоит в том, что в диапазон назначения попадают все данные, которые были выделены в исходных диапазонах. Причем, соответствующие друг другу данные из различных таблиц должны иметь в точности одинаковые заголовки строк и столбцов (с точностью до символа). В этом случае при выделении диапазона назначения достаточно установить рамку выделения в его левый верхний угол, а исходные диапазоны выделяются вместе с заголовками строк и столбцов, по которым распознается соответствие консолидируемых данных. В диалоговом окне консолидации следует установить флажки, определяющие, какие заголовки должны учитываться. Excel, выполняя команду, распознает указанные заголовки в исходных диапазонах и включит их в диапазон назначения, а на пересечении строк и столбцов разместит консолидированные данные, полученные из соответствующих заголовкам строк и столбцов исходных диапазонов. Если в каком-то исходном диапазоне нет строк или столбцов с имеющимися в других исходных диапазонах заголовками, данные из этого листа просто не будут использоваться при вычислении.

Второй способ позволяет консолидировать данные выборочно. Для этого нужно подготовить таблицу, которая будет использоваться в качестве диапазона назначения, создать «шаблон», по которому будет выполняться консолидация. Для этой таблицы необходимо задать заголовки только тех строк и столбцов, данные из которых подлежат консолидации. При использовании этого метода при выделении диапазона назначения необходимо включить в этот диапазон заголовки строк и столбцов, причем они должны быть написаны в точности, как на исходных листах. Исходные диапазоны при выделении тоже должны включать соответствующие заголовки.

Рассмотрим консолидацию по заголовкам строк и столбцов на следующем примере: необходимо получить итоговые данные по расходам за год только для двух филиалов – Северного и Восточного. Таким образом, в диапазон назначения нужно включить только столбцы, содержащие названия филиалов и суммарные расходы и строки, соответствующие только двум названным филиалам. Чтобы выполнить консолидацию, необходимо подготовить диапазон назначения (лист «Выборочные итоги»), включив в таблицу заголовки столбцов «Филиал» (столбец А) и «Всего» (столбец В) и заголовки строк «Северный» (строка 2) и «Восточный» (строка 3), а также строку «Итого:». В качестве диапазона назначения выделяется диапазон A1:B3. В качестве исходных диапазонов выделяются диапазоны A1:E5 на рабочих листах, содержащих отчеты по кварталам. При консолидации устанавливаются флажки, указывающие Excel, что первая строка и столбец представляют собой заголовки консолидируемых данных. Результат консолидации показан на рис. 9.13.

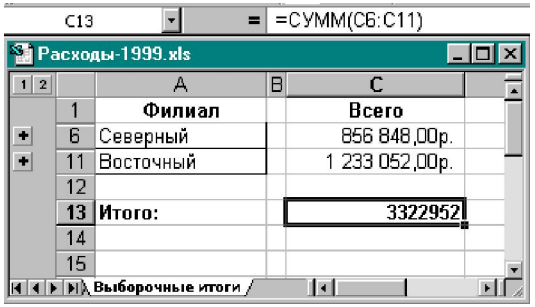


***Рис. 9.13*. Консолидация по заголовкам**

При выполнении команды был установлен флажок «Создавать связи с исходными данными». Установка этого флажка необходима, если исходные данные, использованные при консолидации, могут изменяться. Кроме того, в результате такого способа консолидации информация на рабочем листе автоматически структурируется: с помощью появившихся справа на рабочем листе кнопок или команд меню **Данные** можно раскрыть детальные данные, позволяющие проследить, какая информация

была консолидирована и какие рабочие книги послужили ее источником.

Консолидацию можно выполнить программно в приложении, разработанном на основе Excel. Для описанного выше примера процедура консолидации будет выглядеть следующим образом:

```
Sub Консолидация()
  Dim W As Workbook
  For Each W In Workbooks
    If W.Name = "Расходы-1999.xls" Then
      Windows("Расходы-1999.xls").Activate
      Exit For
    End If
  Next W
  if ActiveWorkbook.Name <> "Расходы-1999.xls" Then
    Workbooks.Open _
                FileName:="D:\Примеры\Расходы-1999.xls"
  End If
  Windows("Расходы-1999.xls").Activate
  Sheets("Выборочные итоги").Select
  Range("A1:B3").Select
  Selection.Consolidate Sources:=Array( _
   "'D:\Примеры\[Расходы-1999.xls]Квартал1'!R1C1:R5C5", _
   "'D:\Примеры\[Расходы-1999.xls]Квартал2'!R1C1:R5C5", _
   "'D:\Примеры\[Расходы-1999.xls]Квартал3'!R1C1:R5C5", _
   "'D:\Примеры\[Расходы-1999.xls]Квартал4'!R1C1:R5C5"), _
     Function:=xlSum, TopRow:=True, _
     LeftColumn:=True, CreateLinks:=True
End Sub
```

Использование именованных диапазонов облегчает ввод ссылок.

9.3.8. Форматирование данных в таблицах

При работе с электронными таблицами бывает необходимо не только ввести табличные данные и выполнить расчеты, но и представить информацию в нужном формате.

При работе с Excel форматирование можно выполнять на уровне страниц, строк и столбцов, отдельных ячеек и символов текста, представляющих значения ячеек.

Для определения *формата страницы* следует выполнить команду **Параметры страницы** меню **Файл**. В открывшемся

диалоговом окне, выбрав соответствующую вкладку, можно определить характеристики страниц (размер бумаги и ориентацию, а также другие свойства), размеры полей, ввести или отменить колонтитулы, выводимые при печати внизу и вверху каждой страницы, задать способ печати содержимого рабочих листов.

Ниже приведена процедура форматирования страницы, с помощью которой устанавливаются следующие параметры: размер бумаги – A4; ориентация – книжная («портрет»); качество печати – 300 точек на дюйм; в центре верхнего колонтитула выводится строка "Пример форматирования страницы"; в нижний колонтитул включается дата (слева), номер страницы (в центре) и время (справа); размеры всех полей – 2 см (размеры переведены в дюймы); отступы колонтитулов – по 1,3 см (переведены в дюймы); таблица при печати центрируется по горизонтали и по вертикали; сетка не печатается; комментарии не печатаются; номер первой страницы определяется автоматически; установлены в качестве сквозных первая строка и первый столбец (если таблица не войдет при печати на одну страницу, они будут выводиться на каждой странице); если таблица разбивается на несколько страниц, то порядок их печати будет «вниз, затем вправо». Текст процедуры можно получить с помощью средств макрогенерации.

```
Sub Параметры_Страницы()
With ActiveSheet.PageSetup
        .PrintTitleRows = "$1:$1"
        .PrintTitleColumns = "$A:$A"
    End With
    ActiveSheet.PageSetup.PrintArea = ""
    With ActiveSheet.PageSetup
        .LeftHeader = ""
        .CenterHeader = "Пример форматирования страницы"
        .RightHeader = ""
        .LeftFooter = "&D"
        .CenterFooter = "&P"
        .RightFooter = "&T"
        .LeftMargin = _
          Application.InchesToPoints(0.78740157480315)
        .RightMargin = _
          Application.InchesToPoints(0.78740157480315)
        .TopMargin = _
          Application.InchesToPoints(0.78740157480315)
```

```
        .BottomMargin = _
          Application.InchesToPoints(0.78740157480315)
        .HeaderMargin = _
          Application.InchesToPoints(0.511811023622047)
        .FooterMargin = _
          Application.InchesToPoints(0.511811023622047)
        .PrintHeadings = False
        .PrintGridlines = False
        .PrintComments = xlPrintNoComments
        .PrintQuality = 300
        .CenterHorizontally = True
        .CenterVertically = True
        .Orientation = xlPortrait
        .Draft = False
        .PaperSize = xlPaperA4
        .FirstPageNumber = xlAutomatic
        .Order = xlDownThenOver
        .BlackAndWhite = False
        .Zoom = 100
    End With
End Sub
```

Excel позволяет вставить разметку страниц в таблицу «принудительно» с помощью команды **Разрыв страницы** меню **Вставка**. Эта команда выполняется программно с помощью оператора

```
ActiveWindow.SelectedSheets.HPageBreaks.Add _
                                Before:=ActiveCell
```

Часть информации можно задать непосредственно с помощью команды **Печать** в меню **Файл**.

С помощью *автоформатирования* автоматически выполняется форматирование данных на рабочем листе. Для его активизации следует выделить таблицу, подлежащую форматированию, и выполнить команду **Автоформат** в меню **Формат**. В открывшемся диалоговом окне можно выбрать из списка подходящий формат, просматривая образцы. Изменить некоторые параметры можно с помощью кнопки **Параметры**. Для программного обращения к этой команде используется оператор

```
Selection.AutoFormat _
  Format:=xlRangeAutoFormatClassic1, Number:=True, _
  Font:= False, Alignment:=True, Border:=True, _
  Pattern:=True, Width:=True
```


который указывает выбранный из списка существующих формат и применяемые параметры форматирования.

Excel предоставляет возможность работы с набором *шрифтов*. Шрифт устанавливается при *форматировании ячейки*. Для установки шрифта следует выделить нужную ячейку или диапазон ячеек. Применить установку шрифта или стиля можно не только ко всей ячейке, но и к отдельному фрагменту текста, размещенного в ячейке. Excel дает также возможность изменить *цвет шрифта*, который используется для ввода и отображения данных (через команду форматирования ячейки и вкладку «Шрифт»).

Если в ячейку вводится большой текстовый фрагмент, то при форматировании ячейки можно включить *режим переноса по словам*, чтобы сформатировать абзац внутри ячейки. В этом режиме текст будет выводиться не в одну строку, а будет формироваться абзац по ширине столбца, при этом высота ячейки автоматически регулируется таким образом, чтобы весь текстовый фрагмент поместился в ячейке.

Для улучшения представления данных, повышения их наглядности можно изменить параметры *выравнивания* (горизонтального и вертикального), а также *ориентацию* текста, *наклон строк*. По умолчанию для ячеек установлены следующие типы горизонтального выравнивания для данных: числа выравниваются по правому краю, текст – по левому краю, логические значения – по центру, ошибки – по центру. Параметры выравнивания можно изменить с помощью соответствующих установок на вкладке «Выравнивание» диалогового окна форматирования ячеек.

При вводе заголовков таблиц или надписей на рабочих листах их приходится размещать в нескольких столбцах и соответственно центрировать. Excel предоставляет для автоматизации этих действий возможность *центрирования* текста *по нескольким выделенным столбцам*. Кроме того, существует возможность *объединения* нескольких соседних ячеек, но используя эту операцию, следует помнить, что объединение ячеек может помешать выполнению некоторых команд при работе с таблицей (например, команд меню **Данные**).

Данные, представленные виде таблицы, в Excel можно оформить на профессиональном уровне и придать им более привлекательный вид с помощью *обрамления и заливки* ячеек. Существуют разные типы обрамления. Они отличаются типом и толщиной линий, цветом. Для выделенных ячеек можно также установить цвет фона (заливки) и узор. Эти установки можно сделать на вкладках «Границы» и «Вид» диалогового окна форматирования ячеек.

Для изменения *формата собственно данных* следует выделить содержащие эти данные ячейки и изменить их формат с помощью команды **Формат ▸ Ячейки**, установив нужные параметры на вкладке «Число» диалогового окна. При форматировании данных сначала выбирается из списка нужный формат числа (тип данных), а затем, если это необходимо для выбранного формата, настраиваются его параметры (например, количество знаков после запятой, присутствие разделителя или названия денежной единицы).

Определить нужные значения параметров форматирования можно с помощью справочной системы или путем записи макроса, при выполнении которого осуществляются все необходимые установки. Например:

```
Sub Формат_ячейки()
Selection.NumberFormat = "h:mm:ss"
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
        .WrapText = False
        .Orientation = 0
        .ShrinkToFit = True
        .MergeCells = False
    End With
    With Selection.Font
        .Name = "Arial Cyr"
        .FontStyle = "обычный"
        .Size = 9
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleDoubleAccounting
        .ColorIndex = xlAutomatic
```



```
    End With
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    Selection.Borders(xlEdgeLeft).LineStyle = xlNone
    Selection.Borders(xlEdgeRight).LineStyle = xlNone
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Interior
        .ColorIndex = 15
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
    End With
    Selection.Locked = True
    Selection.FormulaHidden = False
End Sub
```

Строка "h:mm:ss" в данном примере определяет формат представления времени («часы:минуты:секунды»). Эта строка представляет собой «шаблон», в соответствии с которым данные будут отображаться в ячейке при просмотре рабочего листа. Пользователь может определить *собственные форматы* для ввода специфических данных или для ускорения их поиска (например, формат номеров телефонов, формат преджстаавления денежных сумм).

Для ввода *нового формата* следует выполнить следующие действия:

- выполнить команду **Формат ▸ Ячейки**;
- в диалоговом окне открыть вкладку «Число»;
- из списка «Числовые форматы» выбрать категорию «(все форматы)»;
- в списке «Тип» выбрать код формата, наиболее соответствующий тому формату, который создается;
- в строке ввода исправить формат, приведя его к нужному виду (действия по составлению кода формата описаны ниже);

– щелкнуть на кнопке **OK** для сохранения нового формата.

Ненужные форматы могут быть удалены в том же диалоговом окне (кнопка **Удалить**).

Коды форматов состоят из *последовательных секций* для определения формата *положительных чисел*, *отрицательных чисел*, *нулевых значений* и *текста*.

Секции отделяются друг от друга точкой с запятой (символом ‘;’). Если в коде отсутствует секция, определяющая формат текста, то текст будет вводиться без форматирования. Если код будет состоять только из двух секций, то первая часть будет задавать формат положительных чисел и нулей, а вторая – формат отрицательных чисел. Если код состоит только из одной секции, то все числа будут принимать формат, заданный в ней. Символы, используемые при описании кода, приведены в табл. 9.3.

Например, с помощью шаблона, записанного в строке

```
+# ##0,00_ руб.;[Красный]-# ##0,00_ руб.;[Синий]0,00_ руб.;[Зеленый]"Ошибка"
```

для ячейки можно установить следующий формат: перед положительными и отрицательными числами отображается знак (плюс или минус); значения округляются до двух десятичных знаков после запятой; тысячи в записи числа отделяются разделителем; отрицательные числа показываются красным цветом, а нулевые значения – синим; после чисел через пробел добавляется наименование «руб.»; если вводится значение, которое Excel не распознает как число, зеленым цветом выводится слово "Ошибка".

Применить этот нестандартный формат в программе можно с помощью оператора

```
Selection.NumberFormat = _
  "+#,##0.00_ руб\.;[Red]-#,##0.00_ руб\.;[Blue]0.00_ руб\.;[Green]""Ошибка"""
```

(Эта строка не соответствует той, которая вводится при определении формата в диалоговом окне форматирования ячейки, так как в программе на VBA запись должна соответствовать правилам языка VBA.)

Можно *скопировать формат* отдельной ячейки и присвоить его целому диапазону ячеек. Для этого можно воспользоваться командами **Копировать** и **Специальная вставка** меню **Правка**, но можно также воспользоваться и кнопкой **Формат по образцу** стандартной панели инструментов.

***Таблица 9.3*. Назначение символов в коде формата**

Символ	Назначение
?	Обозначает любую цифру, включая 0. Незначащие нули заменяются пробелами
/	Разделяет числитель и знаменатель дроби
0	Обозначает любую цифру. Незначащие нули не удаляются. Десятичные дроби округляются до указанного в формате количества знаков после запятой
#	Обозначает любую цифру. Незначащие нули удаляются. Десятичные дроби округляются до указанного в формате количества знаков после запятой
Основной	Формат, в котором по умолчанию отображаются числа
, (запятая)	Используются для разделения тысяч
. (десятичная точка)	Определяет позицию десятичной точки. Если число начинается с нуля, слева от десятичной точки вводится нуль
_ (подчеркивание)	Резервирует место для символа, который следует за ним, до ввода этого символа
: p _ + ( )	Появляются в той позиции, в которой они введены в формате
E _ E+ e_ e+	Представляют числа в формате научной (экспоненциальной) записи. Значение справа от E обозначает порядок
%	Умножает число на 100 и представляет его в виде процентного отношения
@	Используется как код формата, указывающий, где должен появляться текст, введенный в ячейку
***символ**	Заполняет место, оставшееся в ячейке, символами, указанными после звездочки
“текст”	Отображает текст, заключенный в кавычки
[цвет]	«Окрашивает» ячейку при вводе данных, в коде формата которых указан, в заданный цвет
\ (обратная косая черта)	Отображает следующий за косой чертой символ

Стиль – это *набор всех параметров форматирования*, одновременно присваиваемых выделенным ячейкам. Удобно ввести свой стиль, если часто используется одно и то же сочетание параметров форматирования (например, сочетание шрифта, размера символов, их начертания, шаблона для ввода данных и т.д.). При работе со стилями при изменении некоторых параметров стиля автоматически будут переформатированы все ячейки всех рабочих листов, отформатированные этим стилем. Для создания нового стиля или изменения существующего предусмотрена команда **Формат ▸ Стиль**.

Для строк и столбцов рабочего листа можно задать их высоту и ширину соответственно. Столбцы и строки рабочего листа можно скрыть (например: при печати отчета можно скрыть столбцы, содержащие промежуточные результаты). Эти действия выполняются через меню **Формат** (команды **Строка** и **Столбец**).

Сгенерировать нужный код можно с помощью средств записи макросов.

При разработке приложений на базе Excel можно использовать и команду условного форматирования. Например, приведенный ниже код устанавливает следующий формат: строка «отлично» отображается красным цветом, «хорошо» – зеленым, «удовлетворительно» – синим, а «неудовлетворительно» отображается цветом, установленным при обычном форматировании ячейки.

```
Sub Условное_форматирование()
  Selection.FormatConditions.Delete
  Selection.FormatConditions.Add _
     Type:=xlCellValue, Operator:=xlEqual, _
     Formula1:="=""отлично"""
  Selection.FormatConditions(1).Font.ColorIndex = 3
  Selection.FormatConditions.Add _
     Type:=xlCellValue, Operator:=xlEqual, _
     Formula1:="=""хорошо"""
  Selection.FormatConditions(2).Font.ColorIndex = 50
  Selection.FormatConditions.Add _
     Type:=xlCellValue, Operator:=xlEqual, _
     Formula1:="=""удовлетворительно"""
  Selection.FormatConditions(3).Font.ColorIndex = 55
End Sub
```


При добавлении каждого нового условия можно задать соответствующие ему параметры форматирования (шрифт для текста, границу и заливку для ячейки).

Установленный формат можно изменять. Например, следующий оператор добавляет двойное подчеркивание к параметрам, соответствующим первому условию:

```
Selection.FormatConditions(1).Font.Underline= _
                                  xlUnderlineStyleDouble
```

Пользователь может задать при определении условного формата различные операции сравнения. Например, в следующем коде устатавливается зеленый фон и граница для ячеек, значения которых попадают в диапазон от 3 до 5:

```
Selection.FormatConditions.Delete
Selection.FormatConditions.Add _
       Type:=xlCellValue, Operator:=xlBetween, _
       Formula1:="3", Formula2:="5"
With Selection.FormatConditions(1).Borders(xlLeft)
       .LineStyle = xlContinuous
       .Weight = xlThin
       .ColorIndex = xlAutomatic
End With
With Selection.FormatConditions(1).Borders(xlRight)
       .LineStyle = xlContinuous
       .Weight = xlThin
       .ColorIndex = xlAutomatic
End With
With Selection.FormatConditions(1).Borders(xlTop)
       .LineStyle = xlContinuous
       .Weight = xlThin
       .ColorIndex = xlAutomatic
End With
With Selection.FormatConditions(1).Borders(xlBottom)
       .LineStyle = xlContinuous
       .Weight = xlThin
       .ColorIndex = xlAutomatic
End With
Selection.FormatConditions(1).Interior.ColorIndex = 4
```

Условия могут налагаться как на значение ячейки, так и на формулу, записанную в ячейку. Код, соответствующий различным условиям и форматам, можно получить с помощью средств макрогенерации. Более подробная информация о средствах условного форматирования содержится в справочной системе.

Если во время работы с таблицей она не отображается на экране целиком и возникает необходимость просматривать параллельно две ее части, удаленные друг от друга, можно разбить окно на две части с помощью *вешки разбивки*, находящейся в верхней части вертикальной полосы прокрутки и правой части горизонтальной полосы. Программным способом установить вешку разбивки на нужном расстоянии от верхней и левой границ окна рабочего листа можно с помощью операторов вида:

```
ActiveWindow.SplitRow = 1
ActiveWindow.SplitColumn = 1
```

Эти операторы «закрепляют» в левой части окна один столбец, а сверху – одну строку, которые могут содержать, например, заголовки.

9.4. Анализ данных и подведение итогов

Наиболее мощными объектами Excel, используемыми для создания компонентов анализа данных в информационных системах являются сводные таблицы (***PivotTables***) и диаграммы (***Charts***).

Приложения для анализа данных, созданные на основе Microsoft Excel, имеют общую фундаментальную архитектуру (рис. 9.14).

В приложениях для анализа данных информация обычно считывается из внешней базы данных (доступ к внешним источникам данных рассматривается ниже) в сводную таблицу или на рабочий лист и потом отображается в виде диаграммы. Далее обработанная информация может быть выведена на печать, включена в отчет и т.п.

Средства доступа к данным с помощью VBA рассматриваются в отдельной главе.

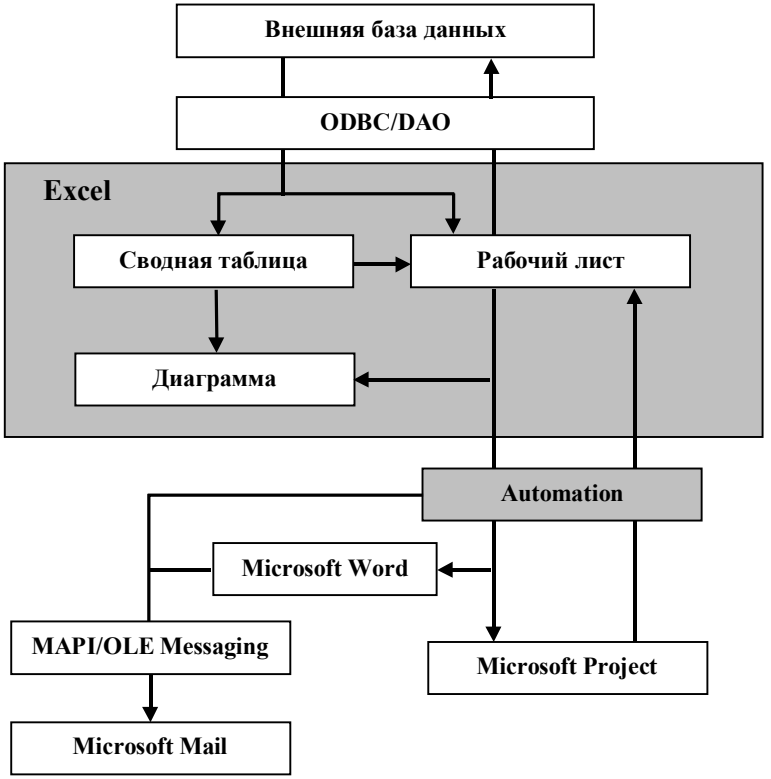


***Рис. 9.14*. Обмен данными в информационных системах Excel**

9.4.1. Создание сводных таблиц

Сводную таблицу можно создать «вручную» с помощью команды **Сводная таблица** из меню **Данные**. Программный способ создания сводной таблицы основан на использовании метода ***PivotTableWizard*** объекта ***Worksheet***. Объект ***PivotTable*** используется только для импорта данных.

Данные в сводной таблице редактировать нельзя, но можно скопировать эти данные из сводной таблицы в диапазон рабочего листа, сделать нужные изменения, экспортировать отредактированные данные в исходную базу данных и выполнить метод ***RefreshTable*** сводной таблицы.

Для создания сводной таблицы средствами VBA, как уже было сказано, необходимо вызвать метод ***PivotTableWizard*** рабочего листа (объекта ***Worksheet***). Этот метод имеет следующие обязательные аргументы:

- ***SourceType*** – признак источника данных: ***xlDatabase*** – список или база данных Excel, ***xlExternal*** – внешняя база данных, ***xlConsolidation*** – консолидация нескольких диапазонов рабочих листов, ***xlPivotTable*** – существующая сводная таблица;
- ***SourceData*** – источник данных, зависящий от первого аргумента: или диапазон, или массив строк, содержащих строку связи ODBC и оператор SQL, или массив диапазонов, или имя существующей сводной таблицы;
- ***TableDestination*** – диапазон, в который будет помещена сводная таблица;
- ***TableName*** – имя, которое будет присвоено сводной таблице;
- ***RowGrand*** – управляет отображением суммарного итога всех строк таблицы (если ***True*** – итог отображается, если ***False*** – нет);
- ***ColumnGrand*** – управляет отображением суммарного итога всех столбцов таблицы (если ***True*** – итог отображается, если ***False*** – нет);
- ***SaveData*** – управление сохранением внутреннего кэша (если ***True*** – содержащий данные сводной таблицы внутренний кэш сохраняется вместе с таблицей в файле рабочей книги, если ***False*** – нет);
- ***HasAutoFormat*** – если этот аргумент равен ***True***, то при изменении данных Excel автоматически переформатирует таблицу;
- ***AutoPage*** – если этот аргумент имеет значение ***True***, Excel автоматически создает поле страницы в сводной таблице (используется только при консолидации диапазонов) ;
- ***Reserved*** – зарезервирован, остается пустым;
- ***OptimizeCache*** – используется для оптимизации создания сводных таблиц с большими и сложными данными;
- ***PageFieldOrder*** – используется для физической ориентации поля страницы на рабочем листе;
- ***PageFieldWrapCount*** – задает номер поля страницы, с которого начинается новая строка или столбец;

- ***ReadData*** – если равен ***True***, то данные сразу считываются в сводный кэш из внешней базы данных, если ***False*** – данные считываются по мере необходимости;
- ***Connection*** – используется для указания источника данных ODBC, источника данных URL или имени файла, содержащего запрос.

При создании сводной таблицы с помощью описанного метода Excel только считывает данные во внутренний кэш и резервирует место для сводной таблицы на рабочем листе. У Excel пока нет информации о том, какие поля нужно помещать в каждую область сводной таблицы (в таблице используются области «Строка», «Столбец», «Данные» и «Страница»), поэтому данные не отображаются на рабочем листе. Таким образом, запросы сводной таблицы действуют на двух уровнях: запрос первого уровня действует для перемещения данных из основного источника данных в кэш сводной таблицы; запросы второго уровня используются для вывода данных из внутреннего кэша в том виде, как они отображаются в таблице. Запросы второго уровня выполняются путем использования свойств и методов объектов ***PivotTable***, ***PivotField*** и ***PivotItem***.

Структура сводной таблицы показана на рис. 9.15.

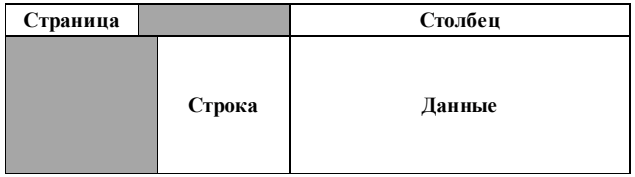


Рис. 9.15. **Упрощенная структура сводной таблицы**

Поле базы данных можно помещать в любую область сводной таблицы. Для этого можно использовать метод ***AddFields*** объекта ***PivotTables*** или присвоить свойству ***Orientation*** объекта ***PivotField*** (поля, включенного в сводную таблицу) одно из приведенных ниже значений констант:

- ***xlColumnField*** – поместить поле в область «Столбец» сводной таблицы;
- ***xlDataField*** – поместить поле в область «Данные»;

– ***xlHidden*** – скрыть поле (поле не отображается ни в каких областях);
– ***xlPageField*** – поместить поле в область «Страница»;
– ***xlRowField*** – поместить поле в область «Строка».

Доступ к отдельному значению можно получить через объект ***PivotItem***.

Ниже приведен пример оператора, создающего на рабочем листе сводную таблицу на основе базы данных Excel:

```
ActiveSheet.PivotTableWizard SourceType:=xlDatabase, _
            SourceData:="Лист1!R1C1:R8C5", _
            TableDestination:="R9C2:R11C2", _
            TableName:="СводнаяТаблица1"
```

Для определения структуры сводной таблицы следует работать уже с созданным объектом ***PivotTable***. Этот объект содержит семейство ***PivotFields***, в котором находятся все поля базы данных, к которым возможен доступ либо по имени поля, либо по его номеру. Используя это семейство, можно помещать различные поля в различные области сводной таблицы. Например:

```
ActiveSheet.PivotTables("СводнаяТаблица1").AddFields _
       RowFields:="Филиал", ColumnFields:="Месяц", _
       PageFields:="Итог"
ActiveSheet.PivotTables("СводнаяТаблица1"). _
    PivotFields("Издержки").Orientation = xlDataField
```

Объект ***PivotTable*** имеет несколько свойств и методов, которые можно использовать для управления структурой таблицы и отображением данных в сводной таблице.

Свойства объекта ***PivotTable***:
– ***ColumnRange*** – диапазон ячеек, включающий область «Столбец» (в том числе столбец заголовков) в сводной таблице;
– ***RowRange*** – диапазон ячеек, включающий область «Строка» (в том числе строку заголовков) в сводной таблице;
– ***PageRange*** – диапазон ячеек, включающий область «Страница» в сводной таблице;
– ***DataBodyRange*** – диапазон ячеек, включающий область «Данные» в сводной таблице;

– ***DataLabelRange*** – диапазон, содержащий имя (имена) поля (полей) сводной таблицы в области «Данные»;
– ***TableRange1*** – диапазон, включающий области «Строка», «Столбец» и «Данные» (но не область «Страница»);
– ***TableRange2*** – диапазон, включающий все области, составляющие сводную таблицу;
– ***ColumnGrand*** – если это свойство установлено в ***True***, отображается итог по столбцам;
– ***RowGrand*** – если это свойство установлено в ***True***, отображается итог по строкам;
– ***HasAutoFormat*** – если это свойство установлено в ***True***, Excel автоматически переформатирует сводную таблицу при ее изменении;
– ***Name*** – имя сводной таблицы;
– ***RefreshDate*** – дата и время последнего обновления данных сводной таблицы в кэше (только для чтения);
– ***RefreshName*** – имя пользователя, который произвел последнее обновление сводной таблицы (только для чтения);
– ***SaveData*** – если это свойство имеет значение ***True***, содержимое внутреннего кэша сохраняется вместе со сводной таблицей;
– ***SourceData*** – источник данных для сводной таблицы (только для чтения);
– ***DisplayErrorString*** – если это свойство установлено в ***True***, сводная таблица выводит значение свойства ***ErrorString*** вместо значений ошибок в ячейках, содержащих ошибки;
– ***ErrorString*** – пользовательская строка с текстом описания ошибки;
– ***DisplayNullString*** – если это свойство установлено в ***True***, сводная таблица выводит значение свойства ***NullString*** в ячейках, содержащих пустые значения;
– ***NullString*** – строка, отображаемая в ячейках, содержащих пустые значения;
– ***ManualUpdate*** – если свойство имеет значение ***True***, изображение сводной таблицы не обновляется при изменениях, что позволяет выполнить последовательность опе-

раторов VBA без обновления изображений на экране (можно отобразить только результат всех изменений);
- ***EnableDrillDown*** – разрешает или запрещает возможность выполнения отображения детальных данных;
- ***EnableFieldDialog*** – разрешает или запрещает пользователю доступ к диалоговому окну «Поле сводной таблицы»;
- ***EnableWizard*** – разрешает или запрещает пользователю доступ к Мастеру сводных таблиц.

Для выполнения некоторых операций над сводной таблицей необходимо выделение диапазонов ее ячеек, что можно выполнить с помощью перечисленных методов. Например, для удаления таблицы нужно выделить весь диапазон содержащих ее ячеек и вызвать метод ***Clear*** для полученного объекта ***Range***. Кроме того, выделенные диапазоны можно отформатировать и т.д.

Последние свойства обеспечивают защиту таблицы.

Методы объекта ***PivotTable***:
- ***AddFields*** – используется для добавления полей в области «Строка», «Столбец» и «Страница» сводной таблицы; для указания области используются аргументы ***RowFields***, ***ColumnFields*** и ***PageFields*** соответственно, их значения могут быть строкой или массивами строк; аргумент ***AddToTable*** принимает значение ***True***, если новые поля нужно добавить к уже существующим в таблице, и ***False***, если указанные поля замещают существующие;
- ***ColumnFields*** – возвращает все поля сводной таблицы в области «Столбец» (в том числе метки столбцов);
- ***RowFields*** – возвращает все поля сводной таблицы в области «Строка» (в том числе метки строк);
- ***DataFields*** – возвращает все поля сводной таблицы в области «Данные»;
- ***PageFields*** – возвращает все поля сводной таблицы в области «Страница»;
- ***PivotFields*** – возвращает все поля сводной таблицы;
- ***VisibleFields*** – возвращает все отображаемые поля сводной таблицы;
- ***HiddenFields*** – возвращает все поля, которые не отображаются в сводной таблице;

- ***CalculatedFields*** – возвращает все вычисляемые поля сводной таблицы;
- ***RefreshTable*** – используется для обновления данных во внутреннем кэше таблицы;
- ***ShowPage*** – используется для создания отдельной сводной таблицы на новом рабочем листе для каждого элемента данных определенного поля из области «Страница»; имя поля указывается с помощью аргумента ***PageField***.

Объект ***PivotField*** применяется для представления полей, содержащих данные, которыми заполняется сводная таблица. Используя свойства и методы этого объекта, можно управлять визуализацией данных в различных областях сводной таблицы.

Свойства полей сводной таблицы в областях «Строка», «Столбец» и «Страница»:
- ***CurrentPage*** – соответствует элементу, отображаемому в поле раскрывающегося списка в области «Страница», применяется только для полей в области «Страница»;
- ***Subtotals*** – предоставляет возможность вычисления промежуточных итогов в сводной таблице; промежуточные итоги могут быть установлены для полей в областях «Строка», «Столбец» и «Страница», но отображаться могут только в областях «Строка» и «Столбец»; это свойство представляет собой массив значений типа ***Boolean***, каждый элемент которого представляет тип используемого промежуточного итога в следующем порядке: «Всего», «Сумма», «Количество значений», «Среднее», «Максимум», «Минимум», «Произведение», «Количество числовых значений»; «Смещенное отклонение», «Несмещенное отклонение», «Смещенная дисперсия», «Несмещенная дисперсия»; таким образом, установив нужные флажки в массиве, можно вычислить интересующие значения промежуточных итогов.

Вычисление промежуточных итогов возможно с помощью свойства ***Function*** метода ***Subtotals***.

Вручную вычисление промежуточных итогов можно задать

с помощью двойного щелчка на метке поля сводной таблицы в диалоговом окне «Вычисление поля сводной таблицы» при работе с мастером сводных таблиц.

Сгруппированные поля имеют некоторые дополнительные свойства. Поля могут группироваться только в областях «Строка», «Столбец» и «Страница».

Данные в поле сводной таблицы можно группировать автоматически или вручную, но оба способа используют метод ***Group***, который является методом объекта ***Range***. Поэтому при вызове этого метода для группировки полей нужно указать диапазон, используя свойство ***DataRange***, применимое ко всем полям сводной таблицы. При автоматической группировке необходимые значения указываются в аргументах метода. Такая группировка может выполняться только с полями сводной таблицы, содержащими данные типа дата/время или числовые данные. Метод ***Group*** имеет следующие аргументы:

- ***Start*** – значение, с которого начинается группа (если установлено в ***True***, то группа начинается с первого значения в поле);
- ***End*** – значение, которым заканчивается группа (значение ***True*** означает, что группа заканчивается на последнем значении в поле);
- ***By*** – единица измерения группы;
- ***Periods*** – состоящий из 7 значений логический массив, с помощью которого производится выбор из семи встроенных интервалов даты/времени («Секунды», «Минуты» и т.д.).

Свойства объекта ***PivotField*** для работы с группами в сводных таблицах (свойства сгруппированных полей):

- ***ChildField*** – возвращает поле, соответствующее прямому подмножеству группы (используется только при ручной группировке);
- ***ParentField*** – возвращает поле, которое содержит в себе указанное поле, т.е. родительское поле (используется только при ручной группировке);
- ***GroupLevel*** – номер уровня поля в группе (существует только у полей, сгруппированных вручную);
- ***TotalLevels*** – общее количество полей в группе.

Свойства полей сводной таблицы в области «Данные»:

- ***NumberFormat*** – задает строку, которая используется для установки формата чисел поля (все числовые форматы можно увидеть на вкладке «Число» диалогового окна «Формат ячейки»; можно использовать пользовательский формат);
- ***Calculation*** – представляет способ визуализации данных (используется для разбиения значений поля в области «Данные»); возможны следующие варианты:
  - ***xlDifferenceFrom*** – «отличие» – отображается модуль разности между значением в поле и заданным значением (***BaseItem***) в заданном поле (***BaseField***);
  - ***xlIndex*** – «индекс» – вычисляется для каждого значения по формуле:

    ((значение в ячейке)*(общий итог))/((итог в строке)*(итог в столбце))

  - ***xlNormal*** – данные отображаются в обычном виде, без дополнительных вычислений;
  - ***xlPercentDifferenceFrom*** – «приведенное отличие» – подобна первому варианту, но отображается процент различия между значением ***BaseItem*** и значениями в поле ***BaseField***;
  - ***xlPercentOf*** – «доля» – отображается процентное соотношение между значением ***BaseItem*** и значениями в поле ***BaseField***;
  - ***xlPercentOfColumn*** – «доля от суммы по столбцу» – значения ячеек области данных отображается в процентах от итога по столбцу;
  - ***xlPercentOfRow*** – «доля от суммы по строке» – значения ячеек области данных отображается в процентах от итога по строке;
  - ***xlPercentOfTotal*** – «доля от общей суммы» – вычисляется процент каждого элемента по отношению к сумме всех значений, взятой за 100%;
  - ***xlRunningTotal*** – «с нарастающим итогом» – значения ячеек области данных отображаются в виде нарастающего итога по строкам;

– ***BaseItem*** – элемент поля сводной таблицы, используемый в свойстве ***Calculation*** с некоторыми установками;
– ***BaseField*** – элемент поля сводной таблицы, используемый в свойстве ***Calculation*** с некоторыми установками;
– ***Function*** – через это свойство Excel предоставляет несколько функций, которые могут применяться к основным данным полей сводной таблицы области «Данные»; может принимать следующие значения:
  - ***xlAverage*** – среднее арифметическое значение анализируемых данных;
  - ***xlCount*** – количество записей;
  - ***xlCountNum*** – количество числовых записей;
  - ***xlMax*** – максимальное значение;
  - ***xlMin*** – минимальное значение;
  - ***xlProduct*** – произведение;
  - ***xlStDev*** – несмещенное стандартное отклонение (по выборке данных);
  - ***xlStDevP*** – смещенное стандартное отклонение (по всей генеральной совокупности);
  - ***xlSum*** – сумма основных данных;
  - ***xlVar*** – несмещенная оценка дисперсии генеральной совокупности (по выборке данных) **;**
  - ***xlVarP*** – смещенная оценка дисперсии (по всей генеральной совокупности).

Применение какой-либо функции к полю приводит к изменению свойства ***Name***. Например: применение функции вычисления среднего к полю “Доход” изменит имя на “Среднее по полю Доход”.

Свойства всех полей сводной таблицы:
– ***DataRange*** – диапазон рабочего листа, занятый изображением данных поля сводной таблицы;
– ***LabelRange*** – диапазон листа, занятый изображением метки имени поля сводной таблицы;
– ***Name*** – имя поля в том виде, как оно изображено в сводной таблице;
– ***SourceName*** – первоначальное имя поля (из источника данных);

– ***Orientation*** – используется для размещения поля в различных областях таблицы; может принимать значения ***xlColumnField***, ***xlDataField***, ***xlHidden***, ***xlPageField***, ***xlRowField***;
– ***Position*** – номер позиции поля в занимаемой им области таблицы;
– ***Value*** – синоним свойства ***Name***;
– ***DataType*** – тип данных поля сводной таблицы; может принимать значения ***xlText*** (текстовые данные), ***xlNumber*** (числовые данные), ***xlDate*** (данные типа дата/время).

Следующие *свойства* объекта ***PivotField*** используются для защиты:
– ***DragToColumn*** – если значение ***False***, поле не может быть перемещено в область «Столбец»;
– ***DragToHide*** – если значение ***False***, поле не может быть скрыто;
– ***DragToPage*** – если значение ***False***, поле не может быть перемещено в область «Страница»;
– ***DragToRow*** – если значение ***False***, поле не может быть перемещено в область «Строка».

Методы объекта ***PivotField*** применяются только к полям, расположенным в областях «Строка», «Столбец» и «Страница»:
– ***PivotItems*** – возвращает имена всех уникальных элементов в поле сводной таблицы;
– ***VisibleItems*** – возвращает элементы, которые видны в данный момент;
– ***HiddenItems*** – возвращает элементы, которые являются скрытыми в данный момент;
– ***ChildItems*** – элементы в поле сводной таблицы, для которых была применена группировка (только для областей «Строка», «Столбец» и «Страница», для полей, сгруппированных вручную);
– ***ParentItems*** – новая группа элементов, которая была сформирована с помощью группировки дочерних элементов (только для областей «Строка», «Столбец» и «Страница», для полей, сгруппированных вручную).

Объект ***PivotItem*** используется для представления уникальных элементов поля сводной таблицы.

Свойства объекта ***PivotItem***:

- ***DataRange*** – объект ***Range***, представляющий диапазон области «Данные», содержащей данный элемент (применяется только для полей в областях «Строка», «Столбец» и «Страница»);
- ***LabelRange*** – объект ***Range***, содержащий метки данного элемента в сводной таблице (применяется только для полей в областях «Строка» и «Столбец»);
- ***Name*** – полное имя элемента поля сводной таблицы или текущее значение, связанное с элементом;
- ***ParetnItem*** – имя группы, к которой принадлежит дочерний элемент (применимо только к элементам, являющимся дочерними в группе; созданной вручную из полей в областях «Строка», «Столбец» и «Страница»);
- ***ShowDetail*** – если имеет значение ***True***, группа отображается полностью (для родительских элементов, созданных при ручной группировке полей в областях «Строка», «Столбец» и «Страница»);
- ***ParentShowDetail*** – если имеет значение ***True***, отображается дополнительная информация о предках элемента, т.е. отображаются все дочерние элементы в соответствующей группе (это свойство дочерних элементов, созданных при ручной группировке полей в областях «Строка», «Столбец» и «Страница»);
- ***Position*** – порядковые номер элемента в указанном поле сводной таблицы (применимо к элементам из областей «Строка», «Столбец», «Страница»);
- ***SourceName*** или ***Value*** – имя элемента в том виде, как оно указано в первоначальном источнике данных;
- ***Visible*** – управляет отображение элемента, позволяет его спрятать без удаления из таблицы (применимо к элементам в областях «Строка», «Столбец» и «Страница»; может принимать значения ***True*** или ***False***).

Метод ***ChildItems*** объекта ***PivotItem*** возвращает элемент или семейство элементов, которые являются дочерними элементами указанного элемента-предка или сгруппированного поля сводной таблицы.

Вычисляемые поля и элементы являются новыми средствами Excel. Это объекты ***PivotFields***, определенные с помощью какой-либо формулы. Например:

```
Activesheet.PivotTables("Сводная таблица1"). _
     CalculatedFields.Add "Прибыль", "=Доход–Издержки"
Activesheet.PivotTables("Сводная таблица1"). _
     PivotFields("Прибыль").Orientation=xlDataField
```

Вычисляемые элементы представляются объектом ***CalculatedItem***. Они имеют *имя* и *формулу*. Вычисляемые поля *вводятся* с помощью вызова метода ***Add*** объекта ***CalculatedFields***, а обрабатываются так же, как и другие объекты ***PivotItems***. Для определения, является ли элемент вычисляемым, можно использовать свойство ***IsCalculated***.

Внутренний кэш сводной таблицы содержит необработанные данные, загруженные из источника данных в оперативную память, а сводная таблица отображает данные кэша. Кэш в программе представляется объектом ***PivotCache***. Он доступен как свойство объекта ***PivotTable***. Этот объект имеет следующие основные свойства:

- ***BackgroundQuery*** – если это свойство установлено в ***True***, Excel выполняет запрос к источнику данных в фоновом режиме, если ***False*** – пользователь должен ждать завершения выполнения запроса перед продолжением работы (в частности, при создании сводной таблицы до размещения полей нужно дождаться завершения запроса);
- ***RecordCount*** – количество записей, находящихся в данный момент в кэше (только чтение);
- ***RefreshDate*** – содержит дату и время последнего обновления кэша;
- ***RefreshName*** – имя пользователя, который произвел последнее обновление кэша;
- ***EnabledRefresh*** – разрешает или запрещает пользователю обновление сводной таблицы.

Далее приведены примеры кода создания и модификации сводной таблицы.

На рис. 9.16 приведена исходная таблица, на основании которой формируется сводная таблица.

	A	B	C	D	E
1	**Филиал**	**Месяц**	**Доходы**	**Издержки**	**Прибыль**
2	Восточный	Янв	$ 10 668,50	$ 9 998,50	$ 670,00
3	Восточный	Фев	$ 12 398,50	$ 11 978,50	$ 420,00
4	Восточный	Мар	$ 12 323,54	$ 12 500,59	$ -177,05
5	Западный	Янв	$ 2 398,50	$ 12 118,50	$ -9 720,00
6	Западный	Фев	$ 22 398,50	$ 12 398,50	$ 10 000,00
7	Западный	Мар	$ 1 208,50	$ 1 098,59	$ 109,91
8	Центральный	Янв	$ 17 398,59	$ 12 233,50	$ 5 165,09
9	Центральный	Фев	$ 12 398,50	$ 12 398,50	$ -
10	Центральный	Мар	$ 1 288,50	$ 1 548,50	$ -260,00
11	**Итого:**		**$ 92 481,63**	**$ 86 273,68**	**$ 6 207,95**
12					

Отчет | Таблица | Лист3 | Лист4

Рис. 9.16. **База данных Excel для создания сводной таблицы**

Следующая процедура создает сводную таблицу:

```
Sub Сводная()
   ActiveSheet.PivotTableWizard _
      SourceType:=xlDatabase, _
      SourceData:="Отчет!R1C1:R10C4", _
                  TableDestination:=_
                 "[Сводная.xls]Таблица!R1C1:R3C1", _
                  TableName:="Сводный отчет", _
                  PageFieldWrapCount:=1
   With ActiveSheet.PivotTables("Сводный отчет")
      .NullString = "-"
      .PageFieldWrapCount = 1
   End With
   ActiveSheet.PivotTables("Сводный отчет"). _
               PivotCache.RefreshOnFileOpen = True
   ActiveSheet.PivotTables("Сводный отчет"). _
               AddFields RowFields:="Данные", _
               PageFields:=Array("Филиал", "Месяц")
   With ActiveSheet.PivotTables("Сводный отчет"). _
                                PivotFields("Доходы")
      .Orientation = xlDataField
      .Position = 1
   End With
   ActiveSheet.PivotTables("Сводный отчет"). _
               PivotFields("Издержки").Orientation _
                                    = xlDataField
End Sub
```

Результат выполнения приведенной выше процедуры показан на рис. 9.17.

Сводная.xls

	A	B	D	E	F
1	Месяц	(Все)	Филиал	(Все)	
2				(Все)	
3	Данные	Всего		Восточны	
4	Сумма по полю Доходы	92481,63		Западный	
5	Сумма по полю Издержки	86273,68		Централь	
6					

Отчет | Таблица | Лист3 | Лист!

Рис. 9.17. **Сводная таблица**

В таблице представлен результат вычислений по собранным в исходной таблице данным. Можно отфильтровать данные по филиалам и по месяцам, выбрав нужный «фильтр» из списков в полях, помещенных в область страницы.

Следующий код позволяет добавить в таблицу вычисляемое поле:

```
Sub ДобавитьВычислимое()
    With Worksheets("Таблица").PivotTables("Сводный отчет")
        .CalculatedFields.Add "Прибыль", "=Доходы-Издержки"
        .PivotFields("Прибыль").Orientation = xlDataField
    End With
End Sub
```

После выполнения данной процедуры в сводную таблицу добавляется еще одна строка, которая позволяет определить прибыль, полученную филиалами за квартал. Приведенные далее процедуры изменяют структуру таблицы и форматируют ее.

```
Sub ИзменитьСтруктуру()
   Worksheets("Таблица").PivotTables("Сводный отчет"). _
      AddFields RowFields:=Array("Филиал", "Данные"), _
                ColumnFields:="Месяц"
End Sub

Sub Форматировать()
   Worksheets("Таблица").PivotTables("Сводный отчет"). _
                         TableRange2.Select
   With Selection.Font
      .Name = "Arial Cyr"
      .Size = 7
      .Strikethrough = False
      .Superscript = False
      .Subscript = False
```

```
    .OutlineFont = False
    .Shadow = False
    .Underline = xlUnderlineStyleNone
    .ColorIndex = xlAutomatic
End With
Selection.EntireColumn.AutoFit
Worksheets("Таблица"). _
          PivotTables("Сводный отчет"). _
                        DataLabelRange.Select
Selection.HorizontalAlignment = xlCenter
With Selection.Interior
    .ColorIndex = 15
    .Pattern = xlSolid
    .PatternColorIndex = xlAutomatic
End With
End Sub
```

Результат последовательного выполнения процедур показан на рис. 9.18.

Сводная.xls

	A	B	C	D	E	F
3			Месяц			
4	Филиал	Данные	Янв	Фев	Мар	Общий итог
5	Восточный	Сумма по полю Доходы	10668,5	12398,5	12323,54	35390,54
6		Сумма по полю Издержки	9998,5	11978,5	12500,59	34477,59
7		Сумма по полю Прибыль	$ 670,00	$ 420,00	$-177,05	$ 912,95
8	Западный	Сумма по полю Доходы	2398,5	22398,5	1208,5	26005,5
9		Сумма по полю Издержки	12118,5	12398,5	1098,59	25615,59
10		Сумма по полю Прибыль	$-9 720,00	$ 10 000,00	$ 109,91	$ 389,91
11	Центральный	Сумма по полю Доходы	17398,59	12398,5	1288,5	31085,59
12		Сумма по полю Издержки	12233,5	12398,5	1548,5	26180,5
13		Сумма по полю Прибыль	$ 5 165,09	$ -	$-260,00	$ 4 905,09
14	Итог Сумма по полю Доходы		30465,59	47195,5	14820,54	92481,63
15	Итог Сумма по полю Издержки		34350,5	36775,5	15147,68	86273,68
16	Итог Сумма по полю Прибыль		$-3 884,91	$ 10 420,00	$-327,14	$ 6 207,95
17						

Отчет | Таблица | Лист3 | Лист5 | Лист4

Рис. 9.18. **Модифицированная сводная таблица**

Таким образом, VBA предоставляет все средства для создания сводных таблиц и управления ими.

9.4.2. Подведение промежуточных итогов

При работе с базами, данных кроме вычислений, связанных с получением итоговых данных, часто возникает необходимость подведения промежуточных итогов (например, при формировании отчетов). Для реализации этой задачи используется команда **Итоги…** в меню **Данные** (метод ***Subtotal*** объекта ***Range***). Эта команда позволяет кроме общего итога, охватывающего все данные, находящиеся в таблице, подвести и промежуточные итоги для отдельных групп записей. Промежуточные итоги автоматически включаются в список (базу) данных, основываясь на изменениях в определенных полях данных. До вызова метода данные должны быть структурированы и должным образом отсортированы.

Рассмотрим пример. В базе данных (таблице Excel) содержится информация о продажах (рис. 9.19).

Итоги.xls

	A	B	C	D	E
1	**Продавец**	**Товар**	**Цена**	**Кол-во**	**Сумма**
2	Иванов	Велосипед	700	3	2100
3	Иванов	Самокат	500	4	2000
4	Петров	Велосипед	700	5	3500
5	Петров	Самокат	500	3	1500
6	Петров	Коляска	900	1	900
7	Сидоров	Велосипед	700	2	1400
8	Сидоров	Коляска	900	1	900
9					

Итоги | База

Рис. 9.19. **База данных продаж**

Требуется получить отчет, который бы включал общее количество проданных товаров и сумму, на которую эти товары были проданы. Причем итоги должны быть подведены как для всех продавцов, так и для каждого продавца отдельно.

Для решения этой задачи необходимо упорядочить записи в таблице по столбцу, значения которого определяют принадлежность к группе (в данном случае – по столбцу «Продавец»). Затем выполняется команда **Итоги…** меню **Данные**. В диалоговом окне этой команды в поле «При каждом изменении в» указыва-

ется столбец, по которому группируются данные (данные должны быть отсортированы именно по этому столбцу); операция, которая должна выполняться при подведении итогов (в данном случае – суммирование); определяются столбцы, по которым подводятся итоги (в нашем примере – столбцы «Кол-во» и «Сумма»), и определяется способ расположения итоговых данных (в данном примере итоговые данные располагаются на одной странице, внизу, т.е. под исходными данными).

Для исполнения команды в программе на VBA следует переместить рамку выделения в базу и выполнить оператор

```
Selection.Subtotal _
   GroupBy:=1, Function:=xlSum, _
   TotalList:=Array(4, 5), Replace:=True, _
   PageBreaks:=False, SummaryBelowData:=True
```

В результате выполнения команды данные группируются и таблица принимает вид, показанный на рис. 9.20.

Итоги.xls

	A	B	C	D	E
1	**Продавец**	**Товар**	**Цена**	**Кол-во**	**Сумма**
2	Иванов	Велосипед	700	3	2100
3	Иванов	Самокат	500	4	2000
4	**Иванов Всего**			**7**	**4100**
5	Петров	Велосипед	700	5	3500
6	Петров	Самокат	500	3	1500
7	Петров	Коляска	900	1	900
8	**Петров Всего**			**9**	**5900**
9	Сидоров	Велосипед	700	2	1400
10	Сидоров	Коляска	900	1	900
11	**Сидоров Всего**			3	2300
12	**Общий итог**			**19**	**12300**

Итоги / База

***Рис. 9.20*. Отчет с промежуточными итогами**

Итоги могут быть также подведены и с помощью сводных таблиц, которые рассматривались выше.

9.4.3. Определение частичных сумм

При подведении итогов часто возникает необходимость выполнения какой-либо операции над данными, удовлетворяющими определенным условиям. В Excel включены средства, позво-

ляющие реализовать такие вычисления. К ним относятся стандартные функции Excel **СУММЕСЛИ** (в заданном диапазоне ячеек суммируются только значения, удовлетворяющие определяемым пользователем условиям), **СЧЕТЕСЛИ** (в указанном диапазоне подсчитывается количество ячеек, значения которых удовлетворяют заданным условиям) и др.

Более мощным средством является Мастер частичных сумм, позволяющий сгенерировать формулу для подсчета суммы значений ячеек столбца, расположенных в строках, ячейки которых удовлетворяют заданным пользователем критериям. Вызывается Мастер через меню **Сервис**. В результате его работы в определяемую пользователем ячейку заносится формула массива, позволяющая получить нужную частичную сумму. Формула для вычисления частичной суммы может быть построена для таблиц, представляющих собой списки (базы данных) Excel.

Рассмотрим пример. В базе данных Excel находится информация о заказах клиентов: фамилия клиента, дата исполнения заказа и стоимость заказа. Требуется определить суммарную стоимость всех заказов со стоимостью свыше 150 руб., которые должны быть выполнены до 10.10.2001.

С помощью мастера частичных сумм для решения этой задачи построена формула массива

```
=СУММ(ЕСЛИ($B$2:$B$5<=ДАТАЗНАЧ("10.10.01");ЕСЛИ($C$2:$C$5>150;$C$2:$C$5;0);0))
```

записанная в ячейку **C7** (рис. 9.21).

={=СУММ(ЕСЛИ($B$2:$B$5<=ДАТАЗНАЧ("10.10.01");ЕСЛИ($C$2:$C$5>150;$C$2:$C$5;0);0))}

Частичные суммы.xls

	A	B	C	D	E	F	G	H
1	**Клиент**	**Дата**	**Сумма**					
2	Иванов	18.10.01	100,00р.					
3	Петров	10.10.01	115,00р.					
4	Сидоров	05.10.01	200,00р.					
5	Яковлев	20.10.01	230,00р.					
6								
7			200,00р.					
8								
9								

Заказы

***Рис.9.21*. Вычисление частичной суммы**

Полученную с помощью Мастера формулу можно модифицировать, если, например, требуется использовать более сложные критерии отбора данных, включающие не только операции сравнения, но и функции и т.п. Например: таблица содержит информацию и среднегодовых температурах (в столбце **A** записан год, а в столбце **B** – температура). Требуется определить максимальную среднегодовую температуру только для високосных лет. Формула массива будет выглядеть следующим образом:

```
=МАКС(ЕСЛИ(ОСТАТ($A$2:$A$11;4)=0;$B$2:$B$11;0))
```

(фигурные скобки появляются при вводе формулы с помощью комбинации *Ctrl+Shift+Enter*).

9.4.4. Создание диаграмм

Объект ***Chart*** предоставляет возможность отображать данные из любого диапазона рабочего листа в виде любого из более чем 70 различных видов диаграмм.

Каждая составляющая диаграммы может управляться или настраиваться с помощью отдельных объектов, содержащихся в объекте ***Chart***. Упрощенная иерархическая схема объекта ***Chart*** показана на рис. 9.4.

Ниже приведен текст процедуры на VBA, создающей новую диаграмму на том же рабочем листе, что и исходные данные (диапазон **A1:E10** рабочего листа **Отчет**), представляемые этой диаграммой.

```
Sub ДобавитьДиаграмму()
   Charts.Add
   ActiveChart.ChartType = xlColumnClustered
   ActiveChart.SetSourceData _
      Source:=Sheets("Отчет").Range("A1:E10"), _
      PlotBy:=xlColumns
   ActiveChart.Location _
      Where:=xlLocationAsObject, Name:="Отчет"
   With ActiveChart
      .HasTitle = True
      .ChartTitle.Characters.Text = "Отчет"
      .Axes(xlCategory, xlPrimary).HasTitle = True
      .Axes(xlCategory, xlPrimary). _
         AxisTitle.Characters.Text = "Филиал/месяц"
      .Axes(xlValue, xlPrimary).HasTitle = True
      .Axes(xlValue, xlPrimary).AxisTitle. _
         Characters.Text = "USD"
   End With
End Sub
```


Более подробно ознакомиться со средствами создания и форматирования диаграмм можно с помощью справочной системы. Кроме того, можно воспользоваться средствами макрогенерации для выявления необходимых свойств и методов объектов, включенных в объект ***Chart***.

9.5. Работа со структурой данных

Структура дает возможность организовать несколько уровней представления данных, степени детализации которых рассчитаны на пользователей с разным уровнем требований к информации. Структура позволяет разворачивать или сворачивать часть элементов рабочего листа для более или менее детального отображения данных.

Возможности структурирования в Excel позволяют создавать до восьми уровней структуры.

Структуры в Excel создаются либо автоматически, либо вручную (если данные в таблице организованы неудачно и Excel не может их структурировать автоматически). В общем случае данные следует располагать в таблицах так, чтобы итоговые строки (с общими данными) располагались под строками с исходными (детальными) данными, а итоговые столбцы – справа от столбцов исходных данных. Для визуального выделения итоговых данных их можно выделить в таблице шрифтом или цветом (для создания структуры это роли не играет).

Excel может автоматически создать структуру диапазона или таблицы. Для этого нужно либо выделить диапазон, если структура создается для отдельного диапазона данных, либо выделить любую ячейку, если структура создается для всего рабочего листа; затем в меню **Данные** выполняется команда **Группа и структура ▸ Создание структуры**. Если данные расположены неудачно и Excel не может разобраться с их структурой, на экран будет выведено соответствующее сообщение.

Если Excel удается выявить структуру данных и выполнить их группировку, то в окне рабочего листа появляется дополнительная информация. Например, если при создании структуры была выполнена группировка данных в строках, слева от имен столбцов таблицы появляются кнопки с номерами уровней данных в созданной структуре. Щелчок по кнопке с номером уровня ведет к отображению данных с первого по заданный уровень включительно. Слева от номеров сгруппированных строк показываются маркеры строк (точки) и вертикальные линии в виде скобок, показывающие, какие строки сгруппированы. Перед итоговой строкой отображается кнопка со значком '–', позволяющая скрыть детальную информацию (т.е. данные сгруппированных строк, если они отображены) или отобразить ее (кнопка с символом '+'). Если информация группировалась по столбцам, дополнительные кнопки появляются над именами столбцов.

Символы структуры можно отображать на экране или можно их скрыть. Режимом отображения управляет флажок на вкладке «Вид» диалогового окна параметров, открываемого соответствующей командой меню **Сервис**. Переключение режимов можно выполнять с помощью комбинации клавиш *Ctrl+8*.

Структуру данных можно удалить с помощью команды **Группа и структура ▸ Удалить структуру**.

Структура данных создается Excel автоматически при выполнении некоторых операций, например, при консолидации данных.

Для ручного создания структуры нужно выполнить следующую последовательность действий:

- в строках или столбцах таблицы выделить ячейки, которые нужно сгруппировать при создании структуры (выделяются ячейки, содержащие исходные данные одного уровня, без итоговых данных для создаваемой группы);
- выполнить команду **Группа и структура ▸ Группировать…** из меню **Данные** и в диалоговом окне команды установить переключатель в положение «строки» или «столбцы» в зависимости от расположения исходных данных.

Описанные шаги повторяются для всех группируемых диапазонов ячеек на всех уровнях создаваемой структуры. Для отмены группировки нужно выделить сгруппированный диапазон и выполнить команду **Группа и структура ▸ Разгруппировать…**

Операции по созданию структуры документа можно выполнять с помощью программного кода на VBA. Например, следующие инструкции создают структуру автоматически, без изменения формата данных:

```
ActiveSheet.Outline.AutomaticStyles = False
Selection.AutoOutline
```

Метод ***AutoOutline*** используется для создания структуры для выделенного диапазона. Синтаксис его вызова следующий:

```
Выражение.AutoOutline
```

где выражение определяет объект типа ***Range***. Если выделенный диапазон представляет единственную ячейку, структура создается для всего листа, новая схема замещает все существующие.

Объект ***Outline*** представляет схему (структуру) рабочего листа. Он позволяет, в частности, управлять отображением структурированной информации на рабочем листе. Например, метод ***ShowLevels***, имеющий синтаксис

```
Выражение.ShowLevels(RowLevels, ColumnLevels)
```

показывает специфицированное количество уровней строк и/или столбцов схемы (структуры). Например, оператор

```
ActiveSheet.Outline.ShowLevels 1
```

отображает только строки, расположенные в структуре активного рабочего листа на первом уровне.

Подробнее свойства и методы данного объекта можно изучить, используя справочную систему VBA.

9.6. Подбор параметров и поиск решения

Подбор параметров и поиск решения используются в том случае, когда необходимо получить конкретное значение в определенной ячейке, значение которой вычисляется по формуле и зависит от значений других ячеек.

Если вычисляемое по формуле значение зависит от единственного параметра, значение которого может изменяться, можно поставить перед Excel задачу *подбора* такого значения этого *па-*

раметра, что результат вычислений в известной ячейке будет равен заданному значению. Если же на результат влияет содержимое нескольких ячеек, значение в каждой из которых может изменяться, или на решение накладываются какие-то условия, требуется найти максимальное или минимальное значение, то Excel должен решить задачу *поиска решения*.

Функции подбора параметра и поиска решения реализованы специальными надстройками Excel. Надстройка доступна, когда она загружена. Просмотреть список надстроек можно с помощью команды **Надстройки…** из меню **Сервис**.

9.6.1. Подбор параметра

При подборе параметра Excel ищет такое его значение, при котором формула, значение которой зависит от этого параметра, даст при вычислении заданный результат в указанной ячейке.

Для решения задачи подбора параметра следует выполнить следующую последовательность шагов:

- выделить ячейку, содержащую формулу, значения параметра которой нужно подобрать;
- выбрать в меню **Сервис** команду **Подбор параметра…** и в диалоговом окне команды задать условия ее выполнения: в поле «Установить в ячейке:» должна находиться ссылка на выделенную на первом шаге ячейку; в поле «Значение:» вводится требуемый результат вычисления по формуле, т.е. значение, которое должно быть получено в результате подбора параметра для заданной формулы; в поле «Изменяя значение ячейки:» вводится ссылка на ячейку, которая содержит изменяемый параметр формулы;
- щелкнуть кнопку **OK** для выполнения команды.

В результате выполнения команды будет выведено окно с результатами работы Excel. Если решение найдено, подобранное значение параметра будет вписано в отведенную для него ячейку.

Подбор параметра можно организовать программным путем, с помощью метода ***GoalSeek***. Вызов этого метода выполняется по правилу:

```
Выражение.GoalSeek(Goal, ChangingCell)
```


где **Выражение** возвращает объект типа ***Range***, представляющий единственную ячейку, содержащую формулу; аргументы определяют заданное значение для результата вычислений по формуле и ячейку, значение которой должно быть подобрано. Например, оператор:

```
Range("B1").GoalSeek Goal:=10, ChangingCell:=Range("A1")
```

подбирает значение параметра в ячейке **A1** так, чтобы результат вычисления по формуле, записанной в ячейке **B1**, был равен 10.

9.6.2. Поиск решения

Функции моделирования и поиска решения более высокого уровня находятся в файле надстройки SOLVER.XLA поиска решения.

При решении задачи поиска решения можно изменять параметры поиска. Для определения условий решения задачи следует выполнить следующую последовательность шагов:

- выделить ячейку, содержащую формулу, для которой решается задача поиска решения, дающего заданное значение для указанной формулы;
- выбрать команду **Поиск решения…** в меню **Сервис**;
- в диалоговом окне команды задать условия ее выполнения:
  - в поле «Установить целевую ячейку:» должно быть введено имя или адрес ячейки, содержащей формулу, значения параметров которой должны быть подобраны;
  - с помощью группы переключателей «Равной:» нужно определить, на какой результат вычислений по указанной выше формуле следует ориентироваться (максимальное значение, минимальное значение или заданное конкретное значение); если выбран переключатель «значению:», то в соответствующем поле вводится это значение, которое и является целью поиска решения;
  - в поле «Изменяя ячейки:» перечисляются ссылки на ячейки, значения которых могут изменяться при поиске нужного решения (диапазон смежных ячеек, ссылки на несмежные ячейки, перечисленные через

символ ‘;’), щелчок по кнопке **Предположить** заставляет Excel автоматически составить список изменяемых ячеек, влияющих на результат;

- с помощью кнопки **Добавить** можно ввести ограничения на значения, подбираемых параметров (щелчок по этой кнопке раскрывает диалоговое окно, в котором можно задать дополнительные условия), сформированные условия отображаются в диалоговом окне, «лишние» условия можно удалить из списка, условия можно также изменить;
- кнопка **Параметры** позволяет задать параметры поиска решения.

Выполнение команды запускается кнопкой **Выполнить**. Результаты поиска решения отображаются в диалоговом окне. Найденное решение можно сохранить в указанных ячейках или восстановить в них исходные значения. Решение можно также сохранить как сценарий (работа с ними рассматривается ниже).

Поиск решения реализуется средствами, описанными в файле надстройки. Для определения всех параметров команды, которые следует задать при написании кода соответствующей инструкции на VBA, можно воспользоваться макрогенератором, записав нужный код «вручную» в режиме записи макроса. Например, следующие операторы

```
SolverOk SetCell:="$B$1", MaxMinVal:=3, _
         ValueOf:="10", ByChange:="$A$1:$A$2"
SolverSolve
```

задают условия поиска решения: параметры подбираются для формулы, записанной в ячейке **B1**, в этой ячейке должно быть установлено конкретное значение (число 10), изменяемые значения находятся в диапазоне **A1:A2**; поиск решения начинается щелчком по кнопке **OK** и найденное решение сохраняется в указанных ячейках.

9.6.3. Использование сценариев

Сценарии позволяют вводить различные варианты наборов значений для рабочего листа. В частности, каждое решение, полученное при определении различных условий и параметров поиска решения, может быть сохранено в качестве отдельного сценария.

С помощью сценариев можно организовать хранение в одной ячейке нескольких значений, принадлежащих различным наборам. Таким образом, *сценарием* является фактически каждое уникальное значение в ячейке или каждый уникальный набор значений для группы ячеек.

Сценарий может быть создан, как было показано выше, путем поиска решения. Но он может быть создан и «вручную». Для этого нужно выполнить команду **Сценарии…** из меню **Сервис**, вызывающую Диспетчер сценариев. Кнопка **Добавить…** в диалоговом окне команды позволяет ввести новый сценарий.

При создании сценария в диалоговом окне «Добавление сценария» вводится его имя и ссылки на ячейки, набор значений которых и определяет сценарий (указываются диапазоны ячеек или адреса ячеек, элементы списка разделяются символом ‘;’). Щелчок по кнопке **OK** открывает новое диалоговое окно, в котором вводятся конкретные значения для указанных изменяемых в сценарии ячеек. Набор ячеек запоминается в сценарии щелчком по кнопке **Добавить** или **OK** (кнопка **Добавить** позволяет продолжить создание новых сценариев, а **OK** возвращает пользователя в диалоговое окно Диспетчера сценариев).

Диспетчер сценариев позволяет изменить и удалить существующие сценарии.

Кнопка **Вывести** переносит заданные в сценарии значения в соответствующие ячейки рабочего листа, что дает возможность отследить изменения, происходящие в таблице при смене сценария.

Диспетчер сценариев дает возможность построить отчет, в котором можно сравнить различные сценарии, заданные в них изменяющиеся значения и полученные для каждого набора результаты.

Для работы со сценариями в VBA используется объект ***Scenario***. Каждый объект ***WorkSheet*** содержит семейство объектов ***Scenarios***. Для добавления сценария в рабочий лист используется метод ***Add***. В качестве аргументов ему передаются: текстовая строка ***Name***, содержащая имя сценария; диапазон ***ChangingCells***, представляющий ячейки, к которым применяется сценарий (из-

меняемые ячейки); одно значение или массив значений, применяемых к изменяемым ячейкам (***Values***); текстовая строка комментария ***Comment***; флажок защиты сценария ***Locked*** (если равен ***True***, изменения запрещены) и флажок ***Hidden***, позволяющий скрыть защищенный сценарий (защита сценария работает, если установлен соответствующий режим защиты рабочего листа).

Для выполнения других операций над сценариями также существуют соответствующие методы (***Show***, ***Delete*** и т.п.). Получить справку по свойствам и методам объекта можно в окне просмотра объектов в редакторе VBA.

9.7. Поиск и отбор данных в таблицах Excel

Часто при работе с большими массивами данных, представленными в виде таблицы, возникает необходимость отбора данных, соответствующих различным критериям. Для решения этой задачи в Excel используются команды подменю **Фильтр** в меню **Данные** (команды **Автофильтр** (метод ***AutoFilter*** объекта ***Range***) и **Расширенный фильтр…** (метод ***AdvancedFilter*** объекта ***Range***)).

Эти средства предназначены для поиска информации в «базе данных» (или списке) Excel. *База данных в Excel* представляет собой таблицу, в которой первая строка содержит *заголовки столбцов* таблицы, а каждая следующая строка – это *запись*, состоящая из упорядоченных *полей*, содержащих *атрибуты* (свойства) одного какого-либо объекта. В такой таблице не должно быть пустых строк и столбцов, не допускается объединение ячеек. Строки, содержащие итоговые данные, обязательно должны отделяться от исходных данных таблицы пустой строкой.

9.7.1. Использование автофильтра

Автофильтр – это наиболее простой способ отбора данных, размещенных в «базе данных». Для вызова автофильтра достаточно установить рамку выделения на ячейку в строке заголовков и выполнить команду **Фильтр ▸ Автофильтр**… в меню **Данные**. Программным способом автофильтр можно активизировать командой

```
Selection.AutoFilter
```


После включения автофильтра пользователь может задать условия отбора информации для каждого столбца, щелкнув по кнопке раскрытия списка, расположенной рядом с назнанием столбца в строке заголовков таблицы (рис. 9.22).

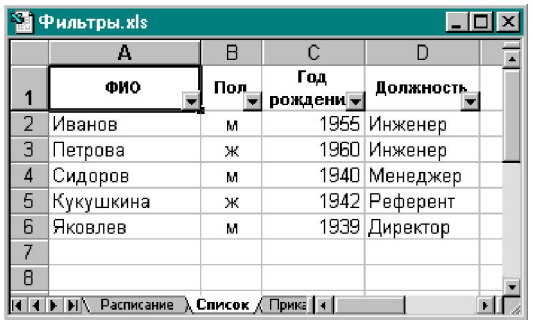


Рис. 9.22. **Установка автофильтра**

Приведенный ниже фрагмент кода позволяет выбрать из всего списка информацию о сотрудниках-мужчинах:

```
Selection.AutoFilter Field:=2, Criteria1:="м"
```

(для второго поля, имеющего в таблице заголовок «Пол», устанавливается условие (критерий) отбора: выбираются записи, содержащие в этом поле значение "м").

Отбор данных можно осуществить сразу по нескольким полям. Например, чтобы получить информацию о мужчинах пенсионного возраста следует к сформированному ранее условию добавить еще одно (задать критерий для поля «Год рождения»):

```
Selection.AutoFilter Field:=2, Criteria1:="м"
Selection.AutoFilter Field:=3, Criteria1:="<=1941"
```

(В таблице после выполнения этого кода останутся всего две строки, содержащие информацию о сотрудниках Сидорове и Яковлеве.)

Отменить условие отбора данных по значению конкретного поля можно, выполнив код

```
Selection.AutoFilter Field:=2
```

(при «ручной» отмене в списке выбирается строка «Все»).

Пользователь может сформировать более сложное условие отбора, используя логические операции «И» и «ИЛИ». Например, вывести информацию о сотрудниках, чьи фамилии начинаются на буквы «И» или «Я», можно с помощью кода

```
Selection.AutoFilter Field:=1, Criteria1:="=И*", _
                     Operator:=xlOr, Criteria2:="=Я*"
```

Но возможности автофильтра ограничены. Рассмотрим, например, следующую задачу: нужно вывести информацию обо всех сотрудниках фирмы пенсионного возраста. Поблема в данном случае состоит в том, что у мужчин и женщин пенсионный возраст различен.

Такую задачу можно решить лишь с помощью расширенного фильтра.

9.7.2. Работа с расширенным фильтром

Для отбора данных с помощью расширенного фильтра необходимо, кроме основной таблицы (*исходного диапазона*, содержащего данные для отбора), сформировать *диапазон условий*, представляющий собой таблицу, в которой строка заголовков содержит те же заголовки, что и исходный диапазон (некоторые заголовки могут быть пропущены, а другие могут повторяться), причем написание заголовков должно в точности совпадать в обеих таблицах, а вместо данных в ячейках этой таблицы должны содержаться условия отбора информации по соответствующим полям.

Условия, сформированные в ячейках одной строки диапазона условий, объединяются при выполнении команды расширенного фильтра логической операцией «И», а условия, размещенные в различных строках – операцией «ИЛИ».

Для поиска данных о сотрудниках пенсионного возраста необходимо сформировать следующие условия в диапазоне условий:

ФИО	Пол	Год рождения	Должность
	м	<=1941	
	ж	<=1946	

Причем столбик «ФИО» и «Должность» можно было бы и опустить, так как значения, находящиеся в этих столбиках в исходном диапазоне, не влияют на условия отбора.

Формировать диапазон условий лучше выше или ниже основной таблицы, но не рядом, так как при срабатывании фильтра строки таблицы, не удовлетворяющие критериям отбора скрываются, а значит, могут оказаться скрытыми и некоторые условия.

Сформировав диапазон условий и установив рамку выделения на ячейку в исходном диапазоне, можно выполнить команду **Фильтр ▸ Расширенный фильтр…** в меню **Данные**. В диалоговом окне команды задаются ее параметры: исходный диапазон и диапазон условий, где должны размещаться результаты отбора (нужно ли их копировать в другое место и куда именно), отбирать ли только уникальные записи (строки) или допускается повторение.

Вызвать расширенный фильтр для приведенного примера можно с помощью команды

```
Range("A1:D6").AdvancedFilter _
     Action:=xlFilterCopy, _
     CriteriaRange:=Range("A10:D12"), _
     CopyToRange:=Range("A15"), Unique:=False
```

(диапазон условий размещается ниже основной таблицы, начиная со строки с номером 10, а результаты копируются на тот же лист, располагаясь, начиная со строки с номером 15).

В результате выполнения команды в диапазоне **A15:D18** автоматически формаруется таблица

ФИО	Пол	Год рождения	Должность
Сидоров	м	1940	Менеджер
Кукушкина	ж	1942	Референт
Яковлев	м	1939	Директор

содержащая записи, удовлетворяющие заданным критериям.

Если бы требовалось выбрать сотрудников, дата рождения которых попадает в заданный диапазон (например с 1940 по 1960 годы), то при формировании диапазона условий пришлось бы повторить столбец «Год рождения», так как два условия, налагаемых на значения поля должны соединяться логической операцией «И»:

Год рождения	Год рождения
>=1940	<=1960

Расширенный фильтр позволяет выполнить отбор информации по сколь угодно сложным критериям.

9.7.3. Поиск данных по подписям строк и столбцов

Мастер поиска Excel может помочь пользователю сформировать формулу для поиска данных по подписям строк и столбцов. В диалоговых окнах Мастера пользователь может задать диапазон, содержащий данные для поиска, а также параметры выбора данных. В результате работы мастера формируется формула поиска вида

=ИНДЕКС($A$1:$D$6; ПОИСКПОЗ($A$13;$A$1:$A$6;); ПОИСКПОЗ($A$14;$A$1:$D$1;))

параметры для поиска можно задавать как константы в формуле или выбирать из ячеек.

В качестве первого параметра функции **ИНДЕКС** указывается диапазон для поиска (в примере это **$A$1:$E$6** – таблица, содержащая данные о сотрудниках фирмы). Второй параметр – номер строки в диапазоне, из которой нужно возвращать значение (может быть пропущен, но тогда обязательным становится третий параметр). Третий параметр – номер столбца, из которого нужно возвращать значение (может быть опущен, если задан номер строки).

В данном примере для определения номеров строки и столбца вызывается еще одна функция – **ПОИСКПОЗ**, которая возвращает относительное положение элемента массива, соответствующего заданному значению указанным образом. При вызове функции в качестве первого ее параметра указывается искомое значение или ссылка на содержащую его ячейку (в нашем примере ячейка **A13** содержит фамилию «Иванов», а ячейка **A14** – строку «Должность»). Второй параметр – диапазон ячеек, возможно, содержащих искомые значения (в нашем примере фамилия «Иванов» ищется в диапазоне **$A1$A6**, а строка «Должность» – в диапазоне **$A$1:$D$1**). Третий параметр задает тип сопоставления, определяющий, как Excel должен выполнять сопоставление искомого значения с элементами массива (возможно точное соответствие, поиск наибольшего значения, которое меньше или равно, чем искомое значение, или наименьшего значения, которое больше или равно искомого значения).

Функция **ПОИСКПОЗ** используется вместо функций типа **ПРОСМОТР**, если нужна позиция элемента в диапазоне, а не сам элемент.

В результате выполнения функции в ячейке, в которую записана показанная формула, вычисляется результат – значение «Инженер» (должность Иванова).

Для поиска данных используются также функции ВПР, ГПР и другие функции ссылки и автоподстановки. Более подробную информацию о работе с функциями можно получить в справочной системе.

9.8. Упорядочение данных в таблицах

Для упорядочения данных, находящихся в базах данных Excel используется команда **Сортировка…** из меню **Данные** (метод ***Sort*** объекта ***Range***).

Для выполнения этой команды можно записать в приложении на VBA оператор

```
Selection.Sort _
  Key1:=Range("A2"), Order1:=xlAscending, _
  Key2:=Range("C2"), Order2:=xlAscending, _
  Header:=xlGuess, _
  OrderCustom:=1, MatchCase:=False, _
  Orientation:=xlTopToBottom
```

Перед выполнением команды рамка выделения должна быть установлена в диапазон, подлежащий сортировке. В приведенном выше примере список сотрудников будет упорядочен по фамилиям (столбец А) в алфавитном порядке; если в списке встретятся однофамильцы, то соответствующие им строки будут упорядочены по году рождения (в порядке убывания); первая строка таблицы рассматривается как строка заголовков (содержит подписи столбцов); при сортировке не учитывается регистр букв; упорядочиваются строки таблицы.

9.9. Защита приложений Excel

При разработке приложений Excel могут возникнуть проблемы, связанные с несанкционированным или случайным изменением данных в рабочих книгах, неправильным вводом. Например, в ячейках таблицы установлены ссылки на ячейки

другого рабочего листа или книги; если после выделения ячейки, содержащей ссылку, ввести любой символ с клавиатуры, связь будет разрушена, нарушится целостность данных, хранящихся в связанных рабочих книгах.

Чтобы избежать такого рода ошибок, следует установить защиту данных, обеспечивающую возможность их изменения только в соответствии с заданными правилами. (Например, нельзя вручную модифицировать данные в ячейках, значения которых вычисляются по формулам.)

9.9.1. Защита от ошибок при вводе данных

Excel позволяет контролировать вводимые пользователем в ячейки рабочего листа даные на соответствие типам ячеек и на принадлежность множеству допустимых для ячейки значений.

При работе с приложением установить защиту (проверку правильности вводимых данных) можно с помощью команды **Проверка…** меню **Данные** (диапазон ячеек, для которых устанавливаются правила ввода данных должен быть предварительно выделен). В диалоговосм окне команды на вкладке «Параметры» можно выбрать тип значений, которые могут содержаться в ячейках выделенного диапазона, и множество допустимых значений (например: диапазон – для чисел, даты, времени; длину – для строки и т.п.). На вкладке «Сообщение для ввода» можно ввести текст сообщения, которое будет служить подсказкой при переходе к вводу данных в ячейку выделенного диапазона. Реакцию на неправильный ввод можно определить с помощью вкладки «Сообщение об ошибке», на которой можно указать вид действия, являющегося реакцией Excel на неправильный ввод («Останов», «Предупреждение» или «Сообщение»), а также заголовок диалогового окна и текст выводимого в случае неправильного ввода ссобщения.

Ниже приведен текст макроса, задающего правила ввода данных в ячейки выделенного перед его выполнением диапазона:

```
Sub Проверка()
  With Selection.Validation
    .Delete
    .Add Type:=xlValidateWholeNumber, _
         AlertStyle:=xlValidAlertStop, _
         Operator:=xlBetween, Formula1:="0", Formula2:="12"
    .IgnoreBlank = False
    .InCellDropdown = True
    .InputTitle = "Количество отработанных часов:"
    .ErrorTitle = "Ошибка ввода"
    .InputMessage = "Допустимы целые числа от 0 до 12"
    .ErrorMessage = _
       "Данные введены неправвильно!" & Chr(10) _
       & "Допустимы только целые числа от 0 до 12!"
    .ShowInput = True
    .ShowError = True
  End With
End Sub
```


Процедуру поверки значений можно сгенерировать с помощью средств макрогенерации.

9.9.2. Защита данных от несанкционированного доступа

Защита данных в Excel устанавливается в несколько шагов. Сначала нужно установить *защиту на уровне ячейки* с помощью команды **Формат ▸ Ячейки…** На вкладке **Защита** диалогового окна данной команды можно установить два флажка, один из которых защищает ячейку от непосредственных изменений, которые может внести пользователь в рабочий лист (запрещает редактирование содержимого ячейки, ее удаление и перемещение), а второй позволяет скрыть формулы, по которым осуществляются вычисления значений ячеек рабочего листа.

Однако атрибуты защиты, установленные для ячеек, не действуют, пока не установлена *защита рабочего листа*. Защитить рабочий лист можно с помощью команды **Защита ▸ Защитить лист…** из меню **Сервис**. В открывшемся диалоговом окне можно установить, какие элементы рабочего листа подлежат защите, а также задать пароль, который позволит ограничить право отмены защиты.

Изменить атрибуты защиты ячейки можно, только сняв защиту с рабочего листа. Снимается защита также с помощью команды меню **Сервис**.

Таким образом, при разработке приложения следует установить защиту ячеек, которые не должны меняться непосредственно на рабочем листе, а должны обновляться только по заданным в приложении правилам (например: при вычислениях или вводиться с помощью пользовательских форм). А с ячеек, данные в которые должны вводиться непосредственно, защиту необходимо снять.

Кроме защиты рабочих листов с помощью команды меню **Сервис**, можно организовать защиту рабочей книги. В диалоговом окне этой команды устанавливается защита структуры рабочей книги (предотвращает удаление, перемещение, вставку и т.п. рабочих листов в книге) и защита окон (предотвращает перемещение, изменение размеров, удаление и т.п. окон).

При работе в сети разработчик может управлять доступом к книге со стороны нескольких пользователей. К книгам Excel можно обеспечить монопольный или совместный доступ. Установить режим доступа к рабочей книге можно с помощью команды **Доступ к книге…**, которая позволяет на вкладке «Правка» включить или отключить флажок «Разрешить совместный доступ»; кроме того, на вкладке «Подробнее», можно уточнить параметры совместного доступа (режим регистрации и внесения изменений).

Однако при совместном доступе к данным возникают проблемы реализации взаимного исключения при работе нескольких пользователей с общими данными, если данные во время работы модифицируются. Эту проблему следует решать программным способом, чтобы обеспечить правильность выполнения транзакций (неделимых единиц работы, которые должны быть выполнены от начала до конца как одна операция), целостность общих данных. Эта проблема решается на этапе проектирования приложения, когда определяется порядок доступа к данным, внесения в них изменений. Для решения проблемы взаимного исключения можно использовать семафорную технику, если средства, предоставляемые Excel, оказываются недостаточными.

Код приложения можно защитить, установив в диалоговом окне свойств VBA-проекта на вкладке «Защита» флажок «Блокировать просмотр проекта». Кроме того, скрыть код приложения можно, сохранив его как надстройку.

Защиту от несанкционированного доступа можно усилить, испльзуя, кроме стандартных средств Excel, процедуры защиты, созданные пользователем-программистом (например, процедуры идентификации пользователей).

Вопросы для самопроверки

1. Определите, что понимается в Excel под рабочей книгой и рабочим листом? Поясните, какие преимущества дает такая организация файлов Excel.
2. Опишите организацию таблиц Excel.
3. Каковы основные возможности электронных таблиц Excel?
4. Каковы общие правила работы с таблицами в Excel?

Иерархия объектов Excel

5. Дайте общую характеристику объектной модели Excel. Какие объекты находятся на верхних уровнях иерархии? Назовите основные объекты. Поясните, какие возможности обеспечивает иерархическая организация объектов.
6. Дайте общую характеристику объекта ***Application***. Перечислите основные его свойства, методы. Какие события этого объекта можно обрабатывать в приложениях на VBA? Какие объекты находятся на следующих уровнях иерерхии?
7. Дайте общую характеристику объекта ***Workbook***. Перечислите основные его свойства, методы. Какие события этого объекта можно обрабатывать в приложениях на VBA? Какие объекты находятся на следующих уровнях иерерхии?
8. Дайте общую характеристику объекта ***Worksheet***. Перечислите основные его свойства, методы. Какие объекты находятся на соседних уровнях иерерхии?
9. Дайте общую характеристику объекта ***Range***. В чем его особенность? Какими способами можно задать этот объ-

ект? Перечислите основные свойства и методы этого объекта.

10. Назовите основные графические объекты Excel. Поясните их назначение, свойства и методы, которые используются при работе с ними.
11. Перечислите основные элементы управления, используемые при создании приложений на основе Excel. Приведите примеры обработки событий для этих элементов.

Представление данных и вычисления в таблицах Excel

12. Поясните варианты ссылок на ячейки рабочих листов Excel. Приведите примеры ссылок на одни и те же ячейки и диапазоны различными способами. Каковы преимущества использования различных способов записи ссылок. В чем отличия? Приведите примеры.
13. Поясните основные правила работы с формулами: что может быть аргументами (операндами) операций, какие операции могут использоваться и каковы их приоритеты, каким образом Excel сообщает об ошибках в формулах, какими способами можно вызвать вычисления по формулам, введенным в ячейки рабочих листов? Приведите примеры.
14. Использование функций: охарактеризуйте все множество функций Excel, опишите правила использования функций. Приведите примеры использования функций для вычислений.
15. Поясните порядок создания и использования пользовательских функций рабочего листа. Приведите примеры.
16. Дайте определение массивов Excel. Какие типы массивов вы знаете? Что такое формула массива и как ее можно задать? Приведите примеры использования массивов.
17. Для чего устанавливаются связи между таблицами? Какие возможности обеспечиваются при создании связей? Каким образом можно определить связи между таблицами? Поясните назначение книги-источника и книги-получалеля. Каким образом обновляются связи?

18. Опишите назначение и варианты консолидации данных в Excel. Приведите примеры реализации различных способов консолидации.
19. Кратко охарактеризуйте возможности форматирования данных в Excel. Опишите средства условного форматирования. Приведите примеры создания пользовательских форматов.

Анализ данных и подведение итогов

20. Дайте определение базы данных в Excel. Какие ограничения налагаются на структуру таблицы, содержащиеся в ней данные? Какие возможности обеспечивает Excel при работе с базами данных? (Воспользуйтесь справочной системой доя получения более полной информации.)
21. Опишите архитектуру средств анализа данных в Excel, их возможностей.
22. Опишите структуру сводных таблиц и порядок их создания. Какие объекты входят в иерархическую схему сводных таблиц? Перечислите основные свойства и методы объектов.
23. Дайте краткое описание средств подведения промежуточных итогов .
24. Опишите средства определения частичных сумм.
25. Опишите иерархическую структуру графических объектов. Какие возможности существуют для создания диаграмм ?

Работа со структурой данных

26. Дайте определение структуры и опишите возможности структурирования данных в Excel.

Подбор параметров и поиск решения

27. Каково назначение средств подбора параметра в Excel? Приведите примеры.
28. Какие возможности обеспечивают средства поиска решения? Опишите способ их использования. Привдите примеры.

29. Дайте определение сценариев. Каким образом можно создать сценарий?

Поиск и отбор данных в таблицах Excel

30. Дайте общую характеристику средств поиска и обора данных в Excel, их возможностей.
31. Поясните использование автофильтра.
32. Опишите порядок работы с расширенным фильтром. Приведите примеры.
33. Опишите средства поиска данных по подписям строк и столбцов.

Упорядочение данных в таблицах

34. Дайте понятие сортировки.
35. Какие ограничения нужно соблюдать при создании таблиц, чтобы обеспечить эффективную работу с ними? приведите примеры.
36. Опишите возможности средств сортировки данных в Excel.

Защита приложений excel

37. Перечислите и проанализируйте проблемы, которые могут возникать при работе с таблицами Excel, их причины.
38. Дайте общую характеристику средств защиты Excel, их назначения и возможностей.
39. Опишите возможности защиты от ошибок при вводе данных, приведите примеры.
40. Опишите средства защиты данных от несанкционированного доступа, порядок их использования.

Глава 10. Разработка приложений с помощью Word

Специалисты в области информационных систем концентрируют усилия на разработке традиционных приложений, связанных с управлением данными, находящимися в базах данных. Задачи, связанные с автоматизацией обработки текстов, ложатся обычно на плечи пользователей текстовых процессоров, несмотря на то что большинство таких задач выполняются с высокой периодичностью.

Средства, предоставляемые текстовыми процессорами, постоянно наращиваются. На основе Microsoft Word могут быть разработаны приложения для решения следующих задач: подготовка публикаций, приложения, преобразующие корпоративные данные в корреспонденцию и обратно, автоматизация делопроизводства на основе шаблонов документов и электронных форм и т.п.

Для решения перечисленных задач могут использоваться следующие возможности, предоставляемые Word и VBA: разработка шаблонов и макросов, добавление новых команд и модификация стандартных команд, разработка вспомогательных средств (меню и панелей инструментов, диалоговых окон). В разрабатываемые шаблоны могут включаться элементы управления и поля, активизирующие создаваемые на их основе документы.

Многие из перечисленных средств являются общими для всех приложений Microsoft Office. В этом разделе более подробно рассматриваются средства, являющиеся специфическими для текстового процессора Word.

Чтобы автоматизировать выполнение задач в Word и добавить дополнительные функции, можно воспользоваться средствами макрогенерации или написать код для решения задач на VBA. Написание кода требует знания модели объектов Word.

10.1. Модель объектов Word

Упрощенная иерархическая структура Word приведена на рис. 10.1. На представленной схеме показаны основные объекты приложения Word, расположенного на самом верхнем уровне иерархии.

Объект ***Application*** представляет Word. Свойство этого объекта ***ActiveDocument*** возвращает ссылку на активный документ, на который установлен фокус. Более подробную информацию можно получить в справочной системе, вызвав справку в окне просмотра объектов редактора VBA после выбора интересующего объекта в списке объектов, его свойства или метода.

При явном вызове объектов указывается точное место объекта в иерархии.

10.1.1. Объект Document

Объект ***Document*** представляет документ Word. Семейство ***Documents*** состоит из всех объектов ***Document***, соответствующих всем открытым документам.

Для активизации конкретного документа из числа открытых используется метод ***Activate***. Для добавления объекта в семейство используется метод ***Add*** (создается новый документ в соответствии с шаблоном и представляющий его объект добавляется в семейство ***Documents***). Например, следующие операторы:

```
Application.Documents.Add _
 Template:="C:\Program Files\Microsoft Office\Шаблоны\Normal.dot", _
   NewTemplate:=False
   MsgBox Application.ActiveDocument.Name
```

создают новый документ на основе шаблона Normal.dot и выводят его имя на экран. Имя или индекс позволяют идентифицировать созданный документ в семействе.

Метод ***Open*** позволяет открыть файл документа Word, представляющий его объект ***Document*** включается в семейство объектов ***Documents***. Например, следующий оператор открывает документ только для чтения:

```
Documents.Open FileName:="C:\MyDoc.doc", ReadOnly:=True
```

Объект ***Document*** имеет иерархическую структуру. Ниже описаны основные, наиболее часто используемые объекты в его структуре.

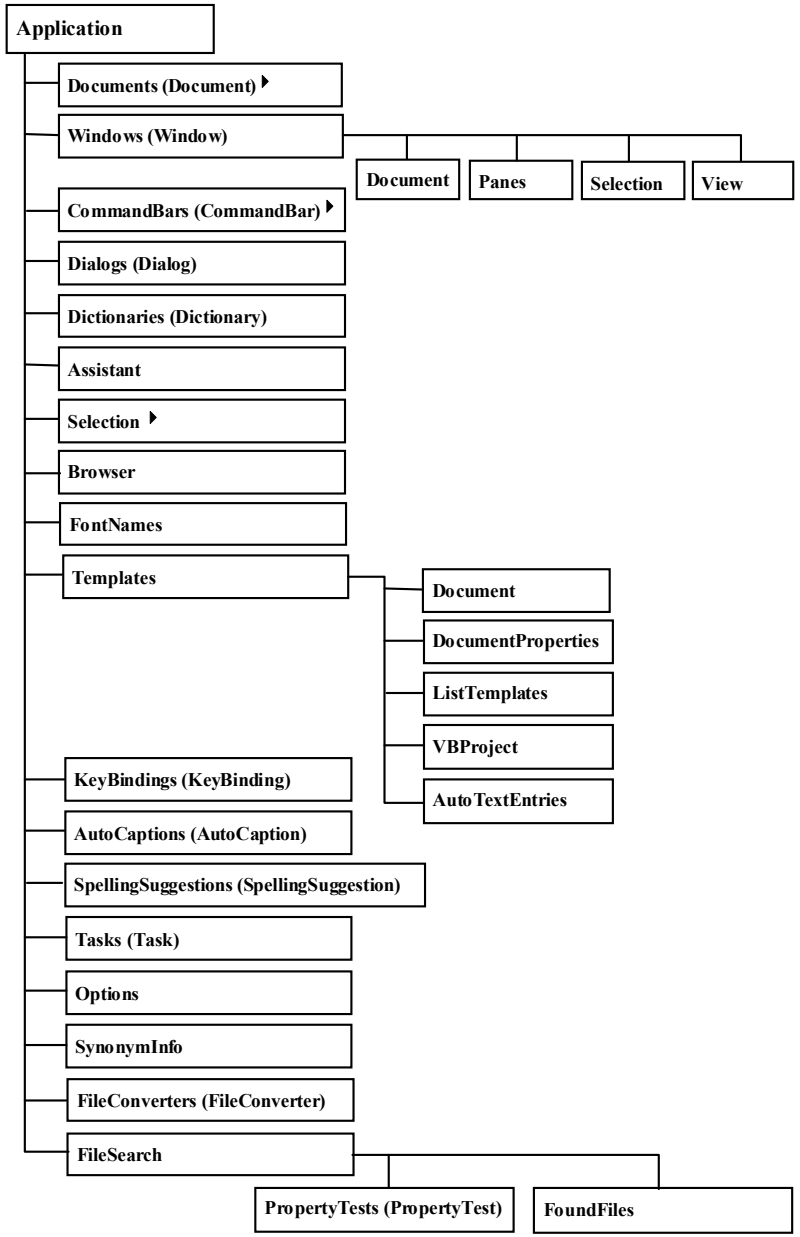


Рис. 10.1. **Упрощенная структура объектов Word**

Свойства объекта ***Document*** представляются семейством ***DocumentProperties***. Каждый объект ***DocumentProperty*** из этого семейства представляет встроенное свойство документа или свойство, заданное пользователем. Список доступных свойств документа можно просмотреть в справочной системе, выбрав в окне просмотра объектов данный объект и вызвав справку. Свойства документа доступны также через обращение к соответствующему свойству объекта ***Document***.

Объект ***Template*** представляет шаблон документа. Объекты этого типа образуют семейство ***Templates***, в которое включены все доступные шаблоны.

Для работы с разделами в документе используется объект ***Section***. Множество разделов документа или выделенной его части представляется объектом-семейством ***Sections***.

Объект ***PageSetup*** представляет описание установок, сделанных для страницы (размер бумаги, поля и т.п.). Установки можно сделать как для всего документа, так и для его раздела. Например, оператор

```
ActiveDocument.PageSetup.RightMargin = InchesToPoints(1)
```

устанавливает размер правого поля равным 1 дюйму.

Объект ***Paragraph*** представляет абзац в тексте документа, а объект ***Paragraphs*** – семейство объектов ***Paragraph***. Эти объекты позволяют включать в текст новые абзацы, форматировать, удалять их и т.п. Следующий оператор выравнивает все абзацы активного документа по правому краю:

```
ActiveDocument.Paragraphs.Format.Alignment = _
                                wdAlignParagraphLeft
```

Семейство объектов ***ListTemplate*** содержит несколько определенных в Word шаблонов маркированных и нумерованных списков.

Семейство ***Lists*** объектов ***List*** содержит множество всех списков заданного документа.

Семейство ***ListParagraphs*** представляет набор объектов ***Paragraph*** заданного документа, списка, диапазона, к которым применено форматирование списков.

Следующий оператор вычисляет количество пунктов в первом списке активного документа:

```
mycount = ActiveDocument.Lists(1).CountNumberedItems
```


Семейство ***Tables*** представляет таблицы в выделенной части документа, заданном диапазоне или во всем документе. Семейство состоит из объектов ***Table***. Объект ***Table*** представляет единственную таблицу. Этот объект структурирован, он имеет иерархическое строение. На следующем уровне иерархии находятся объекты ***Borders***, ***Cells***, ***Columns***, ***Range***, ***Rows*** и ***Shading***.

Следующая последовательность операторов добавляет в документ две пустые таблицы (в начало и в конец документа):

```
Set myRange = ActiveDocument.Range(0, 0)
ActiveDocument.Tables.Add Range:=myRange, NumRows:=3, _
                          NumColumns:=4
Set MyRange = ActiveDocument.Content
MyRange.Collapse Direction:=wdCollapseEnd
ActiveDocument.Tables.Add Range:=MyRange, NumRows:=6, _
                          NumColumns:=10
```

В первой таблице содержится три строки и четыре столбца, а во второй – шесть строк и 10 столбцов.

Объект ***Borders*** – это семейство объектов **Border**, представляющих границы таблицы. Установка свойств этого объекта дает возможность определить характеристики границ.

Объект ***Cells*** – это семейство объектов ***Cell***, представляющих отдельные ячейки таблицы. Эти объекты также имеют «подобъекты», расположенные на следующем уровне иерархии: у каждой ячейки есть границы (***Borders***), каждая ячейка находится в определенном столбце (***Column***) и строке (***Row***), ей соответствует диапазон (непрерывная область в документе – ***Range***), для каждой ячейки можно установить атрибуты заливки («затенения» – ***Shading***).

Следующие операторы выполняют заливку ячейки, находящейся на пересечении первой строки и второго столбца, для заливки используется текстура:

```
Set myCell = _
      ActiveDocument.Tables(1).Cell(Row:=1, Column:=2)
myCell.Shading.Texture = wdTextureAuto
```

Объект ***Columns*** – это семейство объектов ***Column***, представляющих столбцы таблицы. Этот объект имеет дочерние объекты на следующих уровнях иерархии (границы (***Borders***), ячейки,

образующие столбец (***Cells***)) и может иметь свои атрибуты заливки (***Shading***).

Аналогичную структуру имеют объекты ***Row***, представляющие отдельные строки таблицы. Эти объекты объединяются в семейства, представляемые объектами ***Rows***. Каждое семейство – это набор строк в таблице, в выделенном фрагменте или заданном диапазоне.

Семейство ***Words*** представляет набор слов в указанном или выделенном диапазоне или в документе. Каждый элемент этого семейства является объектом ***Range***, представляющим одно слово.

Семейство ***Characters*** – это множество символов в выделенном фрагменте, указанном диапазоне или во всем документе. Каждый элемент этого семейства – объект ***Range***, который представляет один символ.

Объект ***Range*** представляет непрерывную область в документе. Каждый объект ***Range*** определяется позициями начального и конечного символа. Этот объект не зависит от выделения. Он также структурирован: выделение может содержать абзацы, слова, символы и т.д. Следующие операторы выводят количество слов в выделенном фрагменте текста и меняют шрифт первого слова:

```
MsgBox Selection.Words.Count & " words are selected"
With Selection.Words(1)
    .Italic = True
    .Font.Size = 24
End With
```

Семейство объектов ***TableOfContents*** представляет оглавления, вставленные в документ.

Семейство ***Styles*** содержит объекты ***Style***, представляющие как встроенные, так и определенные пользователем стили.

Семейство ***Shapes*** содержит объекты, представляющие графические объекты (работа с этими объектами была описана выше).

Семейство ***InlineShapes*** содержит все графические объекты ***InlineShape***, расположенные в документе в том же слое, что и текст. Таким образом, эти графические объекты превращаются в «символы» текста (данное «превращение» графического объекта

можно выполнить с помощью команды специальной вставки из буфера, сбросив в ее диалоговом окне флажок «Поверх текста»).

Семейство ***Fields*** содержит объекты ***Field***, представляющие поля в указанном диапазоне или выделенном фрагменте документа (работа с полями, их использование в Word подробно рассматриваются ниже).

Семейство ***Comments*** содержит объекты ***Comment***, которые представляют примечания, содержащиеся в выделенном фрагменте, заданном диапазоне или во всем документе. Этот объект структурирован – каждый комментарий (примечание) содержит объект ***Range***.

Объект ***Frame*** представляет рамку, которой может быть очерчен фрагмент текста (например, абзац). Объекты ***Frame*** объединяются в семейства, представляемые объектами ***Frames***. Каждое такое семейство может содержать объекты, соответствующие рамкам в документе, заданном диапазоне или выделенном фрагменте текста. Объект ***Frame*** включает в себя дочерние объекты: ***Borders***, ***Range*** и ***Shading***, т.е. рамка имеет границы, занимает определенный диапазон в тексте документа и для этого диапазона может быть установлена заливка.

Объект ***FormField*** используется для работы с полями форм. Этот объект на следующем уровне иерархии содержит объекты ***CheckBox***, ***TextInput***, ***Range*** и ***DropDown***. Работа с элементами управления подробно рассматривалась выше, при изучении организации пользовательского интерфейса. Объекты этого типа объединяются в семейства – объекты ***FormFields***.

Объекты ***Index*** дают возможность работы с указателями в документах Word. Семейство ***Indexes*** объектов ***Index*** представляет все указатели в заданном документе.

Для создания обычных сносок используются объекты ***Footnote***, а для работы с концевыми сносками в документе предназначены объекты ***Endnote***. Эти объекты также объединяются в семейства.

Еще одним важным объектом, используемым при разработке приложений, являются переменные документа Word – объекты ***Variable***. Переменные документа используются, в частности, для хранения установленных в приложении, в его макросах, значений между вызовами макросов. Переменные могут быть до-

бавлены разработчиком в документ или в шаблон. Все переменные документа объединяются в семейство ***Variables***.

Семейство объектов ***Windows*** содержит объекты ***Window***, представляющие доступные окна.

Перечисленные объекты позволяют вносить изменения в документ. Для изменения характеристик документа используются также свойства объекта ***Document***, его методы и связанные с ним события. Полный их перечень и описание можно получить в справочной системе.

Для сохранения специфицированного документа используется метод ***Save***. Метод ***Close*** закрывает указанный документ.

Объект ***Document*** является одним из основных в модели объектов Word. Ниже рассматриваются другие часто используемые при разработке приложений на основе Word объекты.

10.1.2. Другие дочерние объекты приложения Word

Объект ***Selection*** представляет выделение в окне Word, которое может охватывать как область (диапазон) в документе, так и вырождаться в «точку вставки». Данный объект структурирован: выделение может охватывать произвольный фрагмент текста (диапазон), предложения, отдельные слова, символы, таблицы и их ячейки, графические объекты и т.д. Методы и свойства этого объекта позволяют выполнять форматирование, изменять выделение. Например, следующая инструкция «вырождает» выделение в точку вставки и переносит ее в конец строки:

```
Selection.EndKey Unit:=wdLine, Extend:=wdMove
```

а оператор

```
If Selection.Fields.Count >= 1 Then Selection.Fields.Update
```

обновляет поля (вычисляет их результаты) в выделенной области.

Свойство ***Type*** позволяет получить *тип* выделения (например, «блок» или «точка вставки»). В следующем примере проверяется тип выделения и, если выделение представляет собой только точку вставки, выделяется первый абзац в диапазоне, отмеченном точкой вставки:

```
If Selection.Type = wdSelectionIP Then
        Selection.Paragraphs(1).Range.Select
End If
```


Объекты ***CommandBars*** и ***Dialogs*** рассматривались выше (при изучении интерфейса пользователя).

Объект ***Windows*** – это семейство объектов ***Window***, представляющих доступные окна. Семейство ***Windows*** приложения объединяет все окна Word, а для документа это же семейство содержит только окна этого документа.

Объект ***Dictionaries*** – это семейство объектов ***Dictionary***, представляющих все доступные словари, которые используются для проверки орфографии.

Объект ***KeyBindings*** представляет назначенные пользователем командам клавиатурные сокращения (комбинации клавиш, вызывающие команды). Эти назначения пользователь может сделать на вкладке «Команды» диалогового окна «Настройка», открываемого соответствующей командой меню **Сервис**. Для добавления новой комбинации можно воспользоваться методом ***Add***:

```
CustomizationContext = NormalTemplate
KeyBindings.Add KeyCategory:=wdKeyCategoryCommand, _
  Command:="FileClose", _
  KeyCode:=BuildKeyCode(wdKeyControl, wdKeyAlt, wdKeyC)
```

С помощью приведенного выше кода команде закрытия файла назначается комбинация клавиш *Ctrl+Alt+C*.

Объект ***Templates*** – это семейство шаблонов, доступных приложению.

Объект ***Assistant*** представляет «помощника» Microsoft Office Assistant.

Объект ***Tasks*** – это семейство объектов ***Task***, представляющих все задачи, выполняемые в данный момент в системе. Например, код

```
If Tasks.Exists("Microsoft Excel") Then
     Tasks("Microsoft Excel").Activate
     Tasks("Microsoft Excel").WindowState = _
                              wdWindowStateMaximize
Else
     Shell "C:\MSOffice\excel\Excel.exe"
End If
```

реализует проверку, запущено ли приложение Microsoft Excel; если приложение уже запущено, оно активизируется, его окно

разворачивается на экране; в противном случае осуществляется запуск приложения (загрузка на выполнение программы Excel.exe).

Объект ***Options*** представляет параметры приложения или документа в Word (установленные параметры можно просмотреть в диалоговом окне команды **Параметры** меню **Сервис**).

Объект ***FileConverters*** – это семейство объектов ***FileConverter***, представляющих весь набор конвертеров, которые могут быть использованы при открытии и сохранении фалов. Следующая последовательность операторов позволяет проверить, установлен ли конвертер:

```
For Each conv In FileConverters
      If conv.FormatName = "WordPerfect 6.x" Then
         MsgBox "WordPerfect 6.0 converter is installed"
      End if
Next conv
```

Объект ***FileSearch*** используется для реализации функции поиска в диалоговом окне открытия документа. Он включает в себя два дочерних объекта, представляющих семейство объектов, соответствующих всем критериям поиска, и семейство объектов, представляющих все найденные файлы.

Полный перечень объектов, образующих иерархию объектов Word, можно увидеть, открыв окно просмотра объектов и вызвав справочную систему.

Многие свойства, методы и события, связанные с приложением Word и его объектами сходны со свойствами, методами и событиями приложений Excel, которые рассматривались выше (например ***EnableCancelKey***, ***OnTime***, ***Path***, ***Quit*** и т.п.).

10.2. Работа с полями

Поля в Word представляют собой специальные коды, вставляемые в документ и предназначенные для решения различных задач.

Полем в Word называется специальный набор инструкций, используемых для размещения информации определенного вида в выбранном месте документа. Поля используются обычно для вставки в документ текста или графических изображений, для

обновления информации, изменяющейся по определенным правилам. Коды полей сообщают Word о необходимости вставить в документ на указанное место определенную информацию. Используя поля, можно организовать автоматическое обновление информации или включить режим обновления информации по команде.

Поля появляются в документе при вставке в колонтитул номеров страниц, даты или времени, при создании оглавления, при включении формулы в таблицу и т.д. В Word используются десятки типов полей.

При использовании полей выполняется четыре операции: вставка поля в документ; обновление информации, представленной полем; просмотр полей и переход от одного поля к другому.

Поля динамичны, т.е. их значения изменяются в зависимости от обстоятельств. В полях содержится дополнительная информация, позволяющая выполнять обновление. Поля состоят из трех элементов: символов поля, типа поля и инструкций.

Например, поле даты:

{ Date \@ ММ/д/гг }

Символами поля являются фигурные скобки, обрамляющие собственно поле. Внутри фигурных скобок содержится особый код, указывающий, какая информация должна быть вставлена в документ. Для вставки поля в документ используется команда **Поле…** меню **Вставка**. В открывшемся диалоговом окне можно выбрать категорию и тип поля.

Тип поля задается первым словом в фигурных скобках.

За типом поля следуют *инструкции*. Есть типы полей, которые не требуют специальных инструкций, но в большинстве типов полей инструкции обязательны. По этим инструкциям Word определяет формат вставляемый в документ информации, ее источник.

Поля неявно используются многими командами Word (например, вставка номеров страниц, слияние и т.п.).

По умолчанию после вставки поля в документ на его месте появляется результат этой вставки – *значение поля*. Однако при редактировании документа иногда возникает необходимость просмотра кодов полей. Чтобы сделать код видимым, нужно по-

местить указатель мыши в поле и в контекстном меню, открываемом щелчком правой кнопки мыши, выбрать пункт **Коды/значения полей**. Отмена этого режима осуществляется повторным выполнением той же команды. Для просмотра значения отдельного поля можно также воспользоваться комбинацией клавиш *Shift+F9*.

Если в документе есть множество полей, то можно включить режим просмотра кодов для выделенного фрагмента документа.

В диалоговом окне «Параметры» (вкладка «Вид»), открываемом соответствующей командой меню **Сервис**, можно задать режим отображения полей в документе.

Информация, представляемая некоторыми типами полей, обновляется автоматически (например, номера страниц). Но для многих типов полей необходимо отдавать специальные команды их обновления. В этом случае для обновления значения поля (или полей) нужно выделить поле (или фрагмент текста, содержащий поля) и в контекстном меню выбрать пункт **Обновить поле**.

Иногда бывает необходимо запретить обновление поля. Для этого следует просто *заблокировать* поле с помощью комбинации клавиш *Ctl+F11*, предварительно выделив его. Снять блокировку с поля можно, выделив его и нажав комбинацию клавиш *Ctl+Shift+F11*.

Коды полей можно *напечатать*, задав соответствующий параметр в диалоговом окне печати.

Можно *форматировать* как значение поля, так и его код. Формат представления информации задается с помощью специальных ключей, которые добавляются к коду поля и служат параметрами.

Наиболее часто используемые ключи приведены в табл. 10.1.

Для *перемещения* между полями можно использовать клавишу *F11* или комбинацию клавиш *Alt+F1*. Комбинации клавиш *Shift+F11* или *Alt+Shift+F1* позволяют выполнить переход в обратном порядке.

Таблица 10.1. **Ключи форматирования**

Ключ	Назначение
* caps	Первая буква каждого слова значения поля будет прописной
* firstcap	Первая буква первого слова значения поля будет прописной
* lower	Все буквы в тексте значения поля будут строчными
* upper	Все буквы в тексте значения поля будут прописными
* arabic	Все цифры будут арабскими
* dollartext	Все двузначные числа будут записаны словами, начинающимися с прописных букв. Вместо десятичного разделителя вставляется слово «и», после которого следуют две цифры округленной дробной части числа, записанные арабскими цифрами
* roman	Все цифры будут римскими в нижнем регистре
* Roman	Все цифры будут римскими в верхнем регистре
\@ дддд, MMMM, д, гггг	Дата будет записана словами
* mergeformat	При обновлении сохранит формат значения поля, определенный вручную
* charformat	Формат первой буквы типа поля определяет формат значения поля

Язык VBA предоставляет возможность программного управления полями. Например, для вставки поля, значением которого будет значение переменной документа, в позицию, отмеченную точкой вставки, можно выполнить следующий оператор:

```
Set OFld = Selection.Fields.Add(Range:=Selection.Range, _
   Type:=wdFieldEmpty, Text:="DOCVARIABLE ""Value1""", _
   PreserveFormatting:=True)
```

где переменная **OFld** имеет тип ***Object***.

Для ссылки на поле можно использовать индекс его в семействе объектов. Этот индекс может быть получен после создания поля с помощью оператора

```
FieldIndex = OFld.Index
```

Выделенное поле можно защитить от изменений оператором

```
Selection.Fields.Locked = True
```

а поле, определяемое его индексом, защищает следующая инструкция:

```
ActiveDocument.Fields(I).Locked = True
```

где переменная ***FieldIndex*** содержит значение индекса поля.

Для перехода к следующему/предыдущему полю, для выделения поля, обновления его значения, удаления и т.п. можно использовать свойства и методы объекта ***Field***. Более подробную информацию можно получить в редакторе VBA в окне просмотра объектов, выбрав объекты Word.

Поля удобно использовать для вычислений, при создании шаблонов документов. Для реализации вычислений в документе, сохранения каких-либо значений, используемых в нем, можно использовать переменные документа.

10.3. Переменные документа

В каждом документе определен набор переменных, которые можно добавлять, изменять и использовать с помощью VBA. Эти переменные, в частности, могут использоваться при работе с полями. Следующий пример показывает, как можно добавить переменную документа в активный документ, изменить ее значение и отобразить это значение на экране:

```
Sub AddDocVar()
   With ActiveDocument
      .Variables.Add Name:="DocVar", Value:="1"
      MsgBox .Variables("DocVar").Value
      .Variables("DocVar").Value = _
         .Variables("DocVar").Value + 10
      MsgBox .Variables("DocVar").Value
   End With
End Sub
```

Дополнительную информацию об использовании переменных документа можно получить в справочной системе в окне просмотра объектов.

Переменные документа могут использоваться при включении в текст документа полей, значения которых могут вычисляться. Код поля ***DocVariable*** дает возможность разместить в тексте значение заданной переменной в различных форматах. Например:

```
{ DOCVARIABLE "DocVar" \* DOLLARTEXT }
```

включает в текст значение переменной **DocVar**, причем результат (если значение переменной равно 91) будет отображаться в виде

девяносто один р. 00 коп.

а поле с кодом

```
{ DOCVARIABLE "DocVar" \* ARABIC }
```

покажет просто значение 91 в числовом формате, т.е. строку "**91**".

10.4. Использование формул в документах Word

В документы Word можно вставлять поля, значения которых вычисляются по заданным кодом поля формулам. В этом случае код поля формируется по следующему правилу:

```
{ = Формула [\# "Числовой формат" ] }
```

Для вставки поля с математической формулой в таблицу или основной текст документа можно использовать команду **Формула…** (меню **Таблица**) или команду **Поле…** (меню **Вставка**). В открывшихся диалоговых окнах можно сформировать параметры для кода поля. Причем команда **Формула** предоставляет средства более высокого уровня: доступные для использования в формуле функции, закладки и форматы представления чисел можно выбрать из списков в диалоговом окне команды; а при вставке поля нужно набирать формулу вручную.

В представленном выше синтаксическом правиле записи кода поля, содержащего формулу, используются следующие обозначения: *формула* представляет собой *выражение*, операндами в котором могут быть *числа* или *закладки*, содержащие числа, *поля*, значениями которых являются числа, *ссылки на ячейки таблицы* и *вызовы функций*.

Для определения закладки нужно выделить числовое значение или поле, которое будет поставлено в соответствие закладке,

и выполнить команду **Закладка…** меню **Вставка**. В открывшемся диалоговом окне нужно ввести имя закладки и щелкнуть кнопку **Добавить**. Закладку можно определить для любого выделенного фрагмента текста, в том числе и для таблицы. Существует и программный способ создания закладки – объекта семейства ***Bookmarks*** – с помощью метода ***Add***. Введенные закладки можно использовать не только для поиска соответствующих фрагментов текста, но и для ссылок на эти объекты в формулах. Например, в тексте определены две закладки: «Число» и «Переменная». Первой закладке соответствует строка в тексте, содержащая число, а второй – поле со ссылкой на переменную документа. Тогда следующая формула позволяет сложить значение числа и переменной:

{ = Переменная+Число }

В выражении могут использоваться следующие операции: сложение (+), вычитание (–), умножение (*), деление (/), вычисление процентов (%), степени и корни (^), сравнения на равенство (=), меньше чем (<), меньше либо равно (<=), больше чем (>), больше либо равно (>=), не равно (<>).

В формулах могут использоваться ссылки на следующие функции: ***ABS***(*x*) – абсолютное значение *x*, ***AND***(*x*; *y*) – логическая операция «***И***» для операндов x и y, являющихся логическими выражениями, ***AVERAGE***( ) – среднее значений операндов, включенных в список, ***COUNT***( ) – число элементов в списке, ***DEFINED***(*x*) – определение вычислимости выражения, заданного в качестве операнда (результат равен 1 (истина), если выражение *x* допустимо, или 0 (ложь), если оно не может быть вычислено), ***FALSE*** – 0 (нуль), ***IF***(*x*; *y*; *z*) – условное выражение (возвращает значение *y*, если условие *x* истинно, или *z*, если оно ложно, *y* и *z* могут быть числами или строками), ***INT***(*x*) – целая часть числа или значения выражения *x*, ***MIN***( ) – наименьшее значение в списке, ***MAX***( ) – наибольшее значение в списке, ***MOD***(*x*; *y*) – остаток от деления *x* на *y*, ***NOT***(*x*) – логическое отрицание, ***OR***(*x*; *y*) – логическое «ИЛИ» для операндов *x* и *y*, являющихся логическими выражениями, ***PRODUCT***( ) – произведение значений, включенных в список, ***ROUND***(*x*; *y*) – значение *x*, округленное до указанного десятичного разряда (*y*), ***SIGN***(*x*) – знак числа (1, если *x* > 0, или –1, если *x* < 0), ***SUM***( ) – сумма значений, включенных в список, ***TRUE*** – **1**.

Операндами в выражении, заданном формулой, будут результаты, возвращаемые этими функциями. Для функций с пустыми скобками допустимо любое число аргументов, разделенных точками с запятыми (‘**;**’). Аргументы могут быть числами, формулами или именами закладок. Ссылки на ячейки таблицы допустимы в качестве аргументов следующих функций: ***AVERAGE***(), ***COUNT***(), ***MAX***(), ***MIN***(), ***PRODUCT***() и ***SUM***().

В формулах, вставляемых в ячейки таблицы, можно ссылаться на другие ячейки. Для указания ячейки задаются ее координаты (т.е. название столбца и номер строки). Столбцы обозначаются латинскими буквами (символами ‘A’, ‘B’, ‘C’ и т.д.), а строки нумеруются с 1. При ссылках на ячейки в формулах в качестве разделителя между ссылками на отдельные ячейки используется запятая. Двоеточием разделяются ссылки на первую и последнюю ячейки в диапазонах.

Для организации ссылки на таблицу извне необходимо выделить эту таблицу и создать соответствующую ей закладку. Тогда ссылка на ячейки этой таблицы в формуле будет состоять из имени закладки и ссылки на ячейку или диапазон ячеек, следующей за закладкой через пробел. Например, если в документе есть таблица, состоящая из трех строк и трех столбцов, которая помечена закладкой «Таблица», то для суммирования всех записанных в ее ячейках значений нужно вставить поле с кодом

{ =SUM(Таблица A1:C3) }

Ссылки на ячейки в Word, в отличие от ссылок в Microsoft Excel, всегда являются абсолютными ссылками.

Пример использования таблицы, полей и формул для создания шаблона приведен ниже.

Полученные при вычислении по формуле результаты можно отформатировать, указав нужный формат, т.е. «шаблон» для отображения значений.

Числовой формат указывает способ отображения полученного результата: число знаков в дробной части, способ отображения чисел с различными знаками и т.п.

Числовые форматы, используемые в формулах, сходны с форматами, которые задаются в Excel. В таком шаблоне можно задать формат представления положительных чисел, отрицательных чисел, нулевых значений. Таким образом, общий вид числового формата, используемого в формулах следующий:

“Положительные; Отрицательные; Нулевые”

Числовой формат в формуле может быть опущен или может быть задан в «сокращенном» формате, т.е. в виде “**Положительные**” или “**Положительные; Отрицательные**”. Кроме того, можно определить *текст*, который будет добавляться к вычисленному значению при выводе, и *номер*.

Формат, заданный в виде “**Положительные**” задает общий «шаблон» для вывода всех значений.

Формат “**Положительные**; **Отрицательные**” определяет различные форматы для положительных и отрицательных значений.

Далее описаны элементы числовых форматов, символы, которые могут использоваться в формулах для форматирования результатов вычислений (табл. 10.2).

Простые числовые форматы, которые не содержат ни пробелов, ни текста, необязательно заключать в кавычки, например: **{ ПродажиЗаМарт \# ##0,00 }**. Более сложные числовые форматы, содержащие текст или пробелы, необходимо заключать в кавычки полностью. При вставке поля с помощью команды **Поле** (меню **Вставка**) кавычки добавляются к числовому формату автоматически.

Формулы могут содержать другие поля, например, в следующем фрагменте текста поле, содержащее формулу, вычисляет полное число страниц в документе, нумерация страниц которого начинается с 47:

Стр. { PAGE } из { = (47 - 1) + { NUMPAGES } }

результат изображается в виде «**Стр. 51 из 92**».

Более подробно информацию о работе с полями, таблицами и об использовании формул можно получить в справочной системе.

Таблица 10.2. **Символы форматирования, используемые в формулах**

Элемент	Описание
0 (нуль)	Возвращает цифру (в любом случае). Если полученное значение не содержит цифру в этой позиции, возвращается 0 (нуль). Например, формула **{** = 4 + 5 \# 00,00 **}** возвращает строку «09,00».
#	Возвращает цифру (при наличии). Если полученное значение не содержит цифру в этой позиции, возвращается пробел. Например, формула **{** = 9 + 6 \# ### **}** возвращает строку « 15».
x	Если «**x**» находится слева от десятичного разделителя, цифры, расположенные слева от него, отбрасываются, а если справа – значение округляется до указанного знака. Например: { = 111053 + 111439 \# x## } возвращает строку «492»; { = 1/8 \# 0,00x } возвращает строку «0,125»; { = 3/4 \# ,x } возвращает строку «,8».
, (запятая)	Задает положение десятичного разделителя. Например: **{** = SUM(ABOVE) \# ###,00 **}** возвращает строку «495,47». (При описании формата следует использовать десятичный разделитель, указанный в окне свойств «Язык и стандарты» на Панели управления Windows.)
пробел	Разделяет число на группы по три цифры в каждой. Например: **{** = NetProfit \# "# ### ###" **}** возвращает строку «2 456 800». (При описании формата следует использовать разделитель групп разрядов, указанный в окне свойств «Язык и стандарты» на Панели управления Windows.)
- (минус)	Добавляет минус к отрицательным или пробел ко всем остальным значениям. Например: **{** = 10 – 90 \# -## **}** возвращает «-80».
+ (плюс)	Добавляет плюс к положительным, минус к отрицательным или пробел к нулевым значениям. Например: **{** = 100 – 90 \# +## **}** возвращает строку «+10», а **{** = 90 – 100 \# +## } – «-10».
%, **$**, ***** и т.д.	Включает в значение указанный символ. Например: **{** = Доход \# "##**%**" **}** возвращает строку «33%».
'текст'	Добавляет к значению указанный *текст*. Текст должен быть заключен в апострофы. Например: { = { Цена } *8**,1%** \# "##0,00 **'**является оптовой ценой**'** " } возвращает строку вида «347,44 является оптовой ценой».
`номер`	Добавляет к значению номер предшествующего элемента заданного типа, пронумерованного с помощью команды **Название** (меню **Вставка**) или поля **SEQ**. *Номер* – это идентификатор элемента (например, «Таблица» или «Рисунок»), который должен быть заключен в обратные апострофы (`). Порядковый номер отображается арабскими цифрами. Например: **{** = **SUM**(A1**:**D4) \# "##0,00 ' – общая сумма таблицы' `таблица`" **}** возвращает строку вида «456,34 – общая сумма таблицы 2».

10.5. Использование элементов управления в документах Word

Как уже было сказано, элементы управления используются всеми приложениями Microsoft Office. Они могут включаться в пользовательские формы для создания диалоговых окон, элементы управления можно размещать на рабочих листах Excel или на страницах документа Word.

Панели управления «Формы» и «Элементы управления» в Word выполняют те же функции, что и в Excel, но вид панели «Формы», набор ее инструментов является специфическим для Word (на ней присутствуют кнопки для создания таблиц и рамок).

Средства, предоставляемые панелью «Формы», позволяют включать в документ текстовые поля, поля со списками и флажки. Для каждого такого элемента создается поле соответствующего типа. Однако для разработки приложений лучше использовать элементы управления ActiveX, доступные через панель «Элементы управления». Порядок работы с этими элементами аналогичен способу, описанному для Excel. Программные средства управления элементами также описаны выше. С каждым событием, связанным с элементами управления, можно связать код на VBA, таким образом, значения, которые могут быть введены, выбраны или установлены с помощью этих элементов, можно запомнить в переменных документа и использовать для вычислений, необходимых для создания документа.

Элементы управления, предварительно помещенные в буфер, могут быть «внедрены» в текст с помощью команды **Специальная вставка…** меню **Правка**. Объекты, вставленные с помощью этой команды в текст (в диалоговом окне команды нужно выбрать соответствующий тип объекта, флажок «Поверх текста» должен быть сброшен), превращаются в «символы» в тексте документа, для них можно установить параметры шрифта (например, смещение относительно базовой линии строки).

Элементы управления можно сгруппировать на странице документа в режиме конструктора, соединив их, зафиксировав взаимное расположение элементов, создав с их помощью «форму».

Элементы управления удобно использовать, в частности, для создания шаблонов анкет и т.п. Ниже рассматривается пример такого приложения.

10.6. Создание шаблонов документов на основе таблиц

При создании шаблона документа часто бывает необходимо выполнить его разметку, определив точное положение каждого поля в документе, формат представления данных в нем. Удобным способом разметки документов являются таблицы. Для каждого поля можно отвести отдельную ячейку таблицы, причем обрамление ячеек при необходимости можно снять. Сетка таблицы позволяет точно установить нужные размеры полей с помощью команды **Высота и ширина ячейки…** меню **Таблица**.

Ниже приведен пример шаблона документа, для заполнения которого используются элементы управления (рис. 10.2).

Для ввода фамилии, имени и отчества используются текстовые поля (объекты ***TextBox***).

Рис. 10.2. **Форма для заполнения анкеты в документе Word**

При открытии анкеты в этих полях отображается текст приглашения «Введите…». При перемещении «фокуса» на один из этих элементов весь содержащийся в поле текст выделяется. Для реализации такого выделения должна быть написана процедура обработки события ***GotFocus***, код которой приведен ниже:

```
Private Sub Surname_GotFocus()
   Surname.SelStart = 0
   Surname.SelLength = Len(Surname.Text)
End Sub
```

Код процедуры принадлежит элементу управления Surname (поле для ввода фамилии). Сделанное выделение позволит автоматически стереть весь ранее введенный текст при вводе любого символа, т.е. текст приглашения не нужно стирать вручную.

При переносе фокуса на любое другое поле можно осуществить проверку, не осталось ли поле пустым. Это можно сделать с помощью следующего обработчика события:

```
Private Sub Surname_LostFocus()
    If Surname.Text = "" Then
      Surname.Text = "Введите фамилию"
    End If
End Sub
```

Год рождения – число, которое вводится в текстовое поле. Если требуется ограничить возраст принимаемых на работу, для этого значения можно установить ограничения (в данном примере год рождения должен попасть в диапазон от 1950 до 1975). Кроме того, следует предотвратить возможность ошибки при вводе числа (например, нельзя допустить ввод букв). Для того чтобы можно было не вводить число с клавиатуры, а выбрать значение с помощью мыши, рядом с данным полем размещен элемент управления «Счетчик» (***SpinButton***). В свойствах этого элемента заданы максимальное и минимальное допустимые значения. Значения, соответствующие двум этим элементам, должны быть равны. Ниже приведены обработчики события для текстового поля **ГодРождения**, в которое вводится год рождения, и соответствующего счетчика **AgeCount**, показывающие, как можно контролировать изменение значений элементов управления и установить соответствие между значениями двух элементов.

Код обработчиков:

```
Private Sub ГодРождения_Error(ByVal Number As Integer, _
   ByVal Description As MSForms.ReturnString, _
   ByVal SCode As Long, ByVal Source As String, _
   ByVal HelpFile As String, ByVal HelpContext As Long, _
   ByVal CancelDisplay As MSForms.ReturnBoolean)
   MsgBox "Год рождения должен быть числом от 1950 до 1975"
' В случае ошибки выбирается значение из счетчика:
   ГодРождения.Value = Trim(Str(AgeCount.Value))
End Sub

Private Sub ГодРождения_LostFocus()
    Dim A As Integer
    A = Val(ГодРождения.Value)
    If (A < 1950) Or (A > 1975) Then
' Введено недопустимое значение:
      MsgBox "Год рождения должен быть между 1950 и 1975"
' Восстанавливается прежнее значение:
    ГодРождения.Value = Trim(Str(AgeCount.Value))
    Else
' Значение счетчика устанавливается равным введенному числу:
      AgeCount.Value = A
    End If
End Sub

Private Sub AgeCount_Change()
' Каждое изменение счетчика ведет к изменению поля:
     ГодРождения.Value = Trim(Str(AgeCount.Value))
End Sub
```

Обработчик события через параметры получает дополнительную информацию, которая позволяет более точно определить причину ошибки и организовать ее обработку.

Аналогично можно организовать ввод и контроль дня месяца (числа в дате рождения).

Для ввода месяца в дате рождения используется элемент управления «Поле со списком» (***ComboBox***) **МесяцРождения**. Этот элемент допускает как ввод значения с клавиатуры, так и выбор значения из списка допустимых значений. Включить нужные значения в список можно при открытии документа. Для этого нужно написать обработчик соответствующего события:

```
Private Sub Document_Open()
   With МесяцРождения
        Obj = .AddItem("декабрь", 0)
```

```
        Obj = .AddItem("ноябрь", 0)
        Obj = .AddItem("октябрь", 0)
        Obj = .AddItem("сентябрь", 0)
              ...
    End With
    With Образование
        Obj = .AddItem("начальное", 0)
        Obj = .AddItem("среднее", 0)
              ...
    End With
              ...
End Sub
```

В этой же процедуре можно установить значения всех свойств элементов управления, если они не были установлены при создании объектов.

Аналогично организован выбор значений из списка в поле «Образование» и в полях, в которых указывается степень владения иностранными языками.

Введенную информацию можно сохранить, как документ Word, вывести на печать. Кроме того, информацию, введенную в поля формы, можно сохранить в базе данных для дальнейшего использования. Эти действия можно реализовать в обработчике события закрытия документа.

Если разработанная форма сохраняется в качестве шаблона, на основе которого будут созданы новые документы, то команды инициализации (например, включения элементов в список) должны быть добавлены и в обработчик события создания нового документа.

Для создания законченных приложений необходимо создать свое меню и панели инструментов. Их инициализацию также можно выполнить в обработчике события открытия или создания документа. Ввод более сложно структурированной информации лучше организовать не в тексте документа, а с помощью пользовательских диалоговых окон, которые предоставляют более мощные возможности, в частности, работу с несколькими страницами или вкладками и т.п.

Ввод информации в электронном виде ускоряет и облегчает ее обработку, уменьшает возможность внесения ошибок (принцип «выбирать, а не набирать» гарантирует, что значения будут соответствовать правилам).

Кроме того, электронные формы документов позволяют организовать ввод информации «на расстоянии», по сети. Все продукты Microsoft Office включают средства работы в Internet.

10.7. Защита документов Word и кода

Для документов Word и отдельных их разделов можно установить защиту от изменений их пользователем непосредственно, т.е. изменения в документ могут быть внесены только с помощью средств, предоставляемых пользователю разработчиком приложения (с помощью команд меню, макросов, диалоговых окон и форм с размещенными на них элементами управления).

Для организации защиты содержания документа, его структуры используется команда **Установить защиту…** из меню **Сервис**. В диалоговом окне этой команды можно задать требования по защите документа.

Кнопка **Разделы…** диалогового окна дает возможность определить, к каким разделам будут применяться устанавливаемые правила защиты. Доступ к средствам защиты можно защитить паролем.

После установки защиты в меню **Сервис** появляется команда **Снять защиту**.

Установка защиты автоматически влечет за собой ограничение прав доступа пользователей к командам меню и панелей инструментов. Пользователи получают лишь ограниченные права на выполнение операций в документе (предоставляемые пользователю возможности устанавливаются с помощью переключателя в диалоговом окне).

Для редактирования документа, ввода информации, т.е. для предоставления доступа к некоторым заблокированным объектам в документе при установленной защите требуется использовать специальные средства:

- программный код, который по необходимости снимает и снова устанавливает защиту;

– замену встроенных команд Word пользовательскими командами, добавленными в меню шаблона;
– процедуры, расширяющие или замещающие часть функциональных возможностей Word.

В любом случае с защищенными объектами необходимо работать с помощью программного кода.

Для *установки защиты* документа используется метод ***Protect*** объекта ***Document***. В качестве аргументов при вызове метода передаются значения, представляющие те же настройки, которые можно было выполнить в диалоговом окне команды установки защиты. Правило вызова метода следующее:

```
Документ.Protect (Type, NoReset, Password)
```

Параметр *Type* является обязательным. Он специфицирует тип защиты документа и может принимать следующие значения констант ***WdProtectionType***: ***wdAllowOnlyComments***, ***wdAllowOnlyFormFields***, ***wdAllowOnlyRevisions*** или ***wdNoProtection***.

Аргумент ***NoReset*** является необязательным. Если он равен ***False***, то значения полей в форме принимают значения по умолчанию. Значение ***True*** сохраняет текущие значения полей. Этот аргумент используется только в случае, когда для предыдущего параметра устанавливается значение ***wdAllowOnlyFormFields***, для других типов защиты значение этого аргумента игнорируется.

Параметр ***Password*** не является обязательным. Он требуется, чтобы снять защиту с документа.

Если документ уже защищен, вызов этого метода приведет к ошибке. Для проверки, установлена ли защита и каков ее режим, можно обратиться к свойству ***ProtectionType***. Значением этого свойства может быть значение одной из констант ***WdProtectionType***: ***wdAllowOnlyComments***, ***wdAllowOnlyFormFields***, ***wdAllowOnlyRevisions*** или ***wdNoProtection***. Это свойство доступно только для чтения. Ниже приведен пример проверки защиты и ее установки для документа:

```
If ActiveDocument.ProtectionType = wdNoProtection Then
    ActiveDocument.Protect Type:=wdAllowOnlyComments
End If
```

Этот код устанавливает для активного документа, если он еще не защищен, защиту (пользователю разрешается только работа с примечаниями).

В следующем примере с активного документа, если он защищен, *снимается защита*:

```
If ActiveDocument.ProtectionType <> wdNoProtection Then
       ActiveDocument.Unprotect
End If
```

Прием замещения заблокированных команд состоит в следующем: сначала нужная команда удаляется из меню, а затем добавляется вновь. Этот прием «обманывает» Word, обеспечивая переназначение обработчика недоступной команды. (Работа с панелями команд, меню и панелями инструментов рассматривалась выше.) Вместо заблокированных встроенных команд, можно написать свои процедуры, разместив соответствующие им кнопки на пользовательских панелях инструментов или в пользовательских меню.

Например, следующую процедуру можно использовать для обновления оглавления:

```
Sub ОбновитьОглавление()
      On Error GoTo ErrorHandler
   Dim strMsg As String
   Dim blnVisible As Boolean
   System.Cursor = wdCursorWait
   Application.StatusBar = "Обновление оглавления..."
   Application.ScreenUpdating = False
   blnVisible = ActiveWindow.View.ShowFieldCodes
   If Not blnVisible Then
       ActiveWindow.View.ShowFieldCodes = True
   End If
   Selection.GoTo What:=wdGoToBookmark, _
                 Name:="Оглавление"
   ActiveDocument.TablesOfContents(1).Update
   ActiveDocument.Bookmarks.Add  Name:="Оглавление", _
                                 Range:=Selection.Range
   Selection.MoveUp Unit:=wdParagraph, Count:=2, _
                    Extend:=wdMove
TOCExit:
   On Error Resume Next
   If Not blnVisible Then
         ActiveWindow.View.ShowFieldCodes = False
   End If
   Application.StatusBar = " "
   System.Cursor = wdCursorNormal
   Application.ScreenUpdating = True
   Exit Sub
```

```
ErrorHandler:
   strMsg = "Ошибка: " & Err.Number & vbCrLf
   strMsg = strMsg & "Источник: " & Err.Source & vbCrLf
   Select Case Err.Number
          Case Else
            strMsg = _
"Произошла ошибка в процедуре обновления оглавления" _
                  & strMsg
  End Select
   MsgBox Prompt:=strMsg, Buttons:=vbExclamation, _
          Title:="Ошибка!!!"
   Resume TOCExit
End Sub
```

Эта процедура представляет собой пользовательскую версию заблокированной средствами защиты команды Word. Для поиска оглавления ему назначена закладка.

Для того чтобы эта процедура при выполнении не вызвала ошибку, защиту перед внесением изменений в документ следует снять, как это было показано выше, а перед выходом из процедуры – восстановить первоначальный уровень защиты.

Word предоставляет возможность замещения встроенных команд пользовательскими командами. Для просмотра встроенных команд Word можно воспользоваться командой **Макрос ▸ Макросы** в меню **Сервис**. В раскрывшемся диалоговом окне в списке «Макросы из» нужно выбрать сроку «Команд Word». В представленном выше списке отображаются имена более 900 команд Word, доступных в меню, диалоговых окнах и на панелях инструментов.

Для замены обработчика встроенной команды нужно создать процедуру, присвоив ей то же имя, какое имеет встроенная команда (например, для замены команды сохранения файла следует написать процедуру ***FileSave***). После создания процедуры, имя которой совпадает с именем встроенной команды, Word запускает первую процедуру с известным ему именем, которую находит в следующей последовательности: текущий шаблон, шаблон NORMAL.DOT, загруженные глобальные шаблоны (в алфавитном порядке их имен), встроенные команды Word.

Например, стандартная встроенная процедура создания нового файла (документа или шаблона) выглядела бы так:

```
Sub FileNew()
   Dialogs(wdDialogFileNew).Show
End Sub
```


Эту процедуру можно было бы заменить, добавив в нее код для автоматической установки значений некоторых свойств в диалоговом окне создания документа. Другой способ – полное переопределение процедуры создания документа с введением пользовательских диалоговых окон.

В Excel, PowerPoint и Access приложение (его код) можно защитить, сохранив его как надстройку, но в Word эта возможность отсутствует. Для защиты шаблона в Word в редакторе VBA нужно выделить нужный проект, содержащий защищаемый код, и в контекстном меню или в меню **Сервис** выбрать пункт **Свойства проекта…** В диалоговом окне этой команды на вкладке «Защита» можно установить блокировку просмотра проекта и пароль, позволяющий снять защиту.

Приложение может разрабатываться с использованием нескольких приложений Microsoft Office. Далее рассматриваются средства интеграции различных приложений.

Вопросы для самопроверки

1. Опишите особенности интерфейса Microsoft Office. Какие возможности предоставляют команды меню? Панели инструментов? Какие возможности для настройки имеются в распоряжении пользователей?
2. Какие общие требования к формированию документа Word, позволяющие автоматизировать его создание и обработку, вы можете перечислить? Приведите примеры, связанные с проблемами автоматической генерации документов, вызванные ошибками в подготовке шаблонов документов, их оформлении и форматировании. (Вспомните и перечислите эти требования, изложенные в первой части пособия.)
3. Какие средства автоматизации документа Word Вы знаете? Охарактеризуйте их назначение и возможности. Приведите примеры.
4. Опишите иерархию объектов Microsoft Word. Перечислите основные объекты. Какими свойствами они обладают? Какие методы используются для работы с этими объектами?

5. Опишите объект, представляющий шаблон документа. Каковы возможности использования шаблонов?
6. Каковы особенности объекта ***Range*** в Word? Какова связь этого объекта с другими объектами Word?
7. Каково назначение *полей* в Word? Какие типы полей вы знаете? Каково их назначение и возможности применения для автоматизации документов?
8. Каким образом можно вызвать обновление значений полей?
9. Как просмотреть код поля? Значения и/или коды всех полей в документе?
10. Каковы возможности форматирования полей в Word? Перечислите и опишите основные ключи форматирования, их назначение. Приведите примеры использования.
11. Определите понятие «переменная документа»? Каково назначение переменных документа в Word? Приведите примеры использования переменных документа.
12. Опишите правила записи и использования формул в документе Word. Приведите примеры использования формул в различных ситуациях.
13. Какие функции можно использовать в формулах Word? Приведите примеры использования функций.
14. Опишите правила описания числовых форматов. Какие символы форматирования используются в формулах? Приведите примеры форматирования.
15. Какие элементы управления можно включать в документы Word? Как они используются? Каким образом элементы управления включаются в документ? Каким образом инициализируются элементы управления? Приведите примеры.
16. Охарактеризуйте возможности защиты документов Word и кода. Каким образом можно управлять защитой документа программным способом?

Глава 11. Использование Automation при интегрировании компонентов Microsoft Office

Возможность построения пользовательских приложений на основе существующих программных компонентов позволяет разработчикам создавать новые инструменты и приложения с минимальными затратами.

11.1. Технология ActiveX

Технология ActiveX обеспечивает мощные средства разработки приложений: внедрение объекта из одного приложения в документ, созданный другим приложением, с возможностью его редактирования на месте и Automation, с помощью которого можно программным путем устанавливать свойства, вызывать методы и обрабатывать события как для внедренных объектов, так и для объектов других (внешних) приложений.

ActiveX – это технология, разработанная корпорацией Microsoft, широко применяемая в качестве открытого стандарта, совместного использования данных различными приложениями. Любой разработчик имеет прямой доступ к спецификации (описанию) ActiveX и может создавать приложения, удовлетворяющие требованиям этой спецификации. Приложения могут поддерживать эту технологию в различной степени (например, программный продукт может поддерживать только возможность редактирования объектов на месте, но не поддерживать Automation).

Технологию ActiveX можно представить как механизм, позволяющий рассматривать приложения как набор объектов, которые можно использовать при программировании. Работая с несколькими приложениями, поддерживающими ActiveX, программист имеет дело с несколькими наборами объектов, каждый из которых представляет отдельное приложение. Чтобы из одного приложения можно было получить доступ к другим приложениям посредством ActiveX, оно должно удовлетворять следующим требованиям: приложение должно быть написано в

соответствии со спецификацией ActiveX и приложение должно быть надлежащим образом установлено и зарегистрировано в системном Реестре Windows.

С введением технологии ActiveX фирма Microsoft ввела также новую терминологию программируемых объектов: термин «ActiveX» во многих случаях заменяет термин «OLE» (например: вместо «объект OLE» говорят «компонент ActiveX» или «объект ActiveX», вместо «OLE Automation» – «ActiveX Automation» или просто «Automation»).

Преимущества ActiveX проявляются при объединении в одном проекте наборов объектов, представляющих различные приложения. ActiveX дает возможность разработчикам помещать (или внедрять) объект, созданный в одном приложении, в документ, созданный другим приложением.

11.2. Компонентная модель объектов, внедрение и связывание

Компонентная модель объектов (COM – Component Object Model), являющаяся базовым элементом технологии ActiveX фирмы Microsoft, позволяет пользователям создавать «составные» документы, а разработчикам – «составные» приложения. В составных документах могут содержаться материалы, созданные с помощью различных приложений, а в составных приложениях – различные объекты (из стандартных библиотек или пользовательские), помогающие создать такие материалы. Например, приложение, созданное на основе Access, может использовать объекты Excel в качестве объектов, выполняющих финансовые расчеты.

Automation является одним из наиболее важных средств ActiveX. Это средство позволяет применять в приложении подпрограммы, управляющие объектами в другом приложении. Более ранняя технология организации взаимодействия двух приложений называлась DDE (Dynamic Data Exchange). При использовании DDE приложения вступали для обмена информацией в диалог. Но при таком способе организации взаимодействия приложения работали медленно. Automation является значительно более быстрым и легким в использовании средством.

Ключевым понятием, лежащим в основе всех интегрируемых приложений, является *служба* (*сервис*). Приложение, поддерживающее Automation (OLE Automation), можно рассматривать как набор служб, которыми это приложение может пользоваться. «Специализация» продуктов семейства Microsoft Office такова:

- Access – службы базы данных (СУБД), просмотр и ввод данных, формы, отчеты;
- Excel – службы расчетов, просмотра, финансового анализа и представления данных;
- Word – службы текстового процессора и настольной издательской системы, создание отчетов;
- PowerPoint – службы представления информации, презентации;
- Office Binder – службы хранения и использования данных, интеграции документов;
- Microsoft Outlook – службы управления информацией, электронная почта, управление заданиями, база контактов, планирование групповой работы;
- Project – управление проектами;
- Microsoft Internet Explorer – службы просмотра Web.

Ценой удобства доступа к такому широкому набору служб является производительность. Более высокопроизводительные приложения можно построить с использованием объектов, написанных на языках типа C/C++.

Приложение поддерживает Automation одним из двух способов: может быть *объектом Automation* (*сервер Automation* – это приложение, предоставляющее свои объекты для получения команд от управляющего приложения) и *управляющим приложением* (*клиент Automation* или *контроллер Automation* – это приложение, которое воздействует на объекты Automation и посылает команды серверу Automation).

Некоторые приложения Microsoft Office могут быть только серверами Automation, другие могут выступать и в роли клиентов. Степень поддержки приложениями MS Office Automation показана в табл. 11.1.

Таблица 11.1. **Поддержка Automation приложениями Office**

Приложение	Сервер Automation	Клиент Automation
Access	+	+
Excel	+	+
Office Binder	+	
Outlook	+	
PowerPoint	+	+
Project	+	+
Team Manager	+	
Word	+	+

Технически *объект* в программировании объединяет в себе данные с функциями, использующими эти данные, т.е. в объекте данные объединяются с механизмами доступа к ним. Microsoft расширило понятие объекта до понятия «программируемого объекта». Поведение таких объектов изменяется с помощью Automation. Это позволяет конструировать приложения, управляющие другими объектами-приложениями.

COM определяет интерфейс, позволяющий объектам, написанным разными разработчиками, на разных языках программирования, работать друг с другом. Общий язык программирования для всех продуктов Microsoft – VBA. Он используется для программного управления объектами Automation.

Спецификация COM определяет, как приложения управляют объектами. Возможны два способа совместного использования объектов: пользователи могут *внедрить объекты* других приложений в свои файлы (связать их), что позволяет создавать составные документы, которые содержат данные, порожденные разными приложениями; разработчики могут построить приложения, которые делают общедоступными свои объекты, т.е. разработчики могут *интегрировать существующие функциональные возможности* других продуктов.

OLE (Object Linking and Embedding – связывание и внедрение объектов), как часть спецификации COM, предоставляет

пользователям возможность создания составных документов путем сопоставления объекту двух типов данных: собственно данных объекта и данных для его представления. С помощью OLE пользователи могут связывать или внедрять объекты других приложений.

Когда объекты *связываются* с файлом, они добавляют в него данные для своего представления и указатель на исходный файл объекта. При обновлении исходного файла обновляется и его представление в составном документе. При изменении расположения исходного файла относительно составного документа связи разрушаются.

При *внедрении* объектов в файл в него вносятся и данные о представлении объекта, и данные, необходимые для его редактирования внутри составного документа. Таким образом, все данные, созданные разными приложениями, содержатся в одном документе (файле). Это приводит к «раздуванию» файла составного документа.

Спецификация COM включает в себя часть, называемую Automation, которая позволяет выполнять разработчикам следующие действия:

- управлять с помощью программного кода объектами, созданными другими приложениями;
- создавать и использовать другие пользовательские объекты, поддерживающие спецификацию COM;
- управлять с помощью программного кода связыванием и внедрением объектов.

Приложения, которые управляют доступными объектами других приложений, называются контроллерами Automation (или клиентами Automation), а приложения, которые предоставляют такие объекты в качестве «строительных блоков», – серверами Automation.

Для создания пользовательских объектов используются языки программирования (например, C/C++), а для использования этих объектов, предоставляемых прикладными программами в распоряжение других программ, можно воспользоваться любым языком программирования, обеспечивающим поддержку контроллеров Automation.

При проектировании интегрированного приложения можно выбрать одну из трех моделей:

- *Модель центрального приложения* – одно из приложений Office управляет остальными. Разработчики пишут код только для управляющего приложения.
- *Модель исполняемого файла* – все компоненты приложений Office управляются из самостоятельного EXE-файла. Кроме его программирования, разработчики могут писать код и для приложений подключаемых компонентов. «Главное» приложение может быть написано на любом языке программирования (например, Visual Basic).
- *Модель распределения ролей* – каждое приложение Office и исполняемый EXE-файл разработчиков выполняют свою часть работы, код исполняется внутри «родного» приложения. При необходимости приложение само активизирует нужный компонент.

11.3. Создание объектов Automation

Можно считать, что Automation использует две основные функции: ***CreateObject*** и ***GetObject***.

Функцию создания объекта ***CreateObject*** можно использовать для запуска приложений как серверов Automation этого объекта и создания экземпляра класса объекта-приложения. Объект Automation – это экземпляр или реализация некоторого класса. Использование этой функции подчиняется следующим правилам:

```
Dim ОбъектнаяПеременная As Object
Set ОбъектнаяПеременная = CreateObject( _
                "ИмяПриложения.ТипОбъекта")
```

Соответствие основных приложений Microsoft Office, объектов и классов представлено в табл. 11.2.

При работе с Excel следует использовать класс ***Excel.Application***, если нужно представить данные и оставить пользователя в Excel. Если свойство ***Visible*** этого класса сделать равным ***True***, то можно выйти из приложения, вызвавшего Excel, не закрывая сам Excel. Класс ***Excel.Sheet*** используется, если необ-

ходимо произвести промежуточные расчеты или заполнить рабочий лист данными для последующего использования. При выходе из приложения этот объект будет закрыт. Класс ***Excel.Chart*** используется для создания диаграммы.

В различных версиях MS Office используются различные версии библиотек. Проблемы, связанные с этим, рассматриваются ниже.

При работе с Excel следует использовать класс ***Excel.Application***, если нужно представить данные и оставить пользователя в Excel. Если свойство ***Visible*** этого класса сделать равным ***True***, то можно выйти из приложения, вызвавшего Excel, не закрывая сам Excel. Класс ***Excel.Sheet*** используется, если необходимо произвести промежуточные расчеты или заполнить рабочий лист данными для последующего использования. При выходе из приложения этот объект будет закрыт. Класс ***Excel.Chart*** используется для создания диаграммы.

В различных версиях MS Office используются различные версии библиотек. Проблемы, связанные с этим, рассматриваются ниже.

Таблица 11.2. **Приложения, объекты и классы MS Office**

Приложение	Тип Объекта	Класс	Библиотека
Excel	Приложение	Excel.Application	Microsoft Excel Object Library
	Лист	Excel.Sheet	
	Диаграмма	Excel.Chart	
Word	Приложение	Word.Application	Microsoft Word Object Library
PowerPoint	Приложение	PowerPoint.Application	Microsoft PowerPoint Object Library
Access	Приложение	Access.Application	Microsoft Access Object Library
Office Binder	Редактор связей	Binder	Microsoft Binder Object Library
Outlook	Приложение	Outlook.Application	Microsoft Outlook Object Model

При объявлении переменной типа ***Object*** происходит позднее связывание. Раннее связывание производится на этапе компиляции, если тип переменной-объекта известен заранее. Перед написанием кода, использующего Automation, необходимо сослаться на библиотеку серверов Automation. Ссылку можно установить с помощью команды **Ссылки** меню **Сервис**. После этого можно использовать синтаксис для раннего связывания:

```
Dim ОбъектнаяПеременная As Класс
Set ОбъектнаяПеременная = CreateObject( _
                          "ИмяПриложения.ТипОбъекта")
```

Существует несколько способов загрузки объектов-приложений. Первый метод использует переменную, объявленную объектом нужного типа на уровне процедуры. Это позволит не заботиться об уничтожении объекта по окончании процедуры: так как область видимости объекта ограничена только данной процедурой, закрытие объекта произойдет автоматически при выходе из процедуры. Следует учитывать, что скрытые объекты работают быстрее, чем видимые, поэтому устанавливать свойство ***Visible*** в ***True*** следует только тогда, когда появляется необходимость представить пользователю результаты работы.

Пример кода для загрузки приложения Excel, использующий функцию ***CreateObject***:

```
Sub OLE_Excel_1()
     Dim strMsg As String
     Dim objXL As Excel.Application
     Set objXL = CreateObject("Excel.Application")
     strMsg = "Нажмите Ctrl+Alt_Del, " _
          & " чтобы в диспетчере задач увидеть," _
          & " что запущен Excel (его окно невидимо)"
     MsgBox  Prompt := strMsg, _
             Buttons := vbInformation, _
             Title := "Информация о запуске Excel"
End Sub
```

После завершения процедуры можно снова проверить в Диспетчере задач наличие Excel – приложение будет закрыто.

Другой способ загрузки объекта-приложения основан на использовании переменной, объявленной на уровне модуля:

```
Sub OLE_Excel_2()
   Dim strMsg As String
```



```
   Set obj_XL = CreateObject("Excel.Application")
   obj_XL.Visible = True
   strMsg = "Нажмите ОК для закрытия Excel"
   MsgBox  Prompt := strMsg, _
           Buttons := vbInformation, _
           Title := "Информация о запуске Excel"
   obj_XL.Quit
   Set obj_XL = Nothing
End Sub
```

Здесь описание

```
Dim obj_XL As Excel.Application
```

выполнено на уровне модуля. Объект не закрывается автоматически при завершении процедуры, следовательно, его нужно закрыть с помощью ***Quit*** и установить объектную переменную в исходное состояние.

В программе можно управлять и другими свойствами открытого объекта-приложения. Например, оператор

```
If obj_XL.WindowState <> xlMaximized Then
   obj_XL.WindowState = xlMaximized
End If
```

разворачивает окно объекта-приложения после его запуска на весь экран, если оно еще не было развернуто.

Приведенный ниже код создает экземпляр класса ***Excel.Sheet*** рабочего листа Excel, затем рабочий лист заполняется данными из таблицы базы данных Access и полученный результат сохраняется для дальнейшего использования:

```
Sub MakeXLObject()
  On Error GoTo ErrorMakeXLObject
  ' Объявления переменных:
  Dim dbSolution As Database
  Dim tdfCustomers As TableDef
  Dim rstCustomers As Recordset
  Dim intRow As Integer
  Dim intColumn As Integer
  Dim strSavedReport As String
  Dim strMsg As String
  Dim ws As Workspace
' Переменная objELSheet объявлена на уровне модуля как
'              Dim objXLSheet As Object
' Установка ссылок на объекты:
   Set ws = DBEngine.CreateWorkspace("MyWS", _
```

```
            "Admin", "", dbUseJet)
   Set dbSolution = ws.OpenDatabase("C:\DB\Ole.mdb")
   Set tdfCustomers = dbSolution.TableDefs("Customers")
   Set rstCustomers = dbSolution.OpenRecordset("Customers")
' Установка полного пути и имени для сохраняемого
' листа-объекта Excel (т.е. для отчета):
   strSavedReport = "C:\Samples\OfficeSolutionsRpt.xls"
' Создание объекта - рабочего листа Excel:
   Set objXLSheet = CreateObject("Excel.Sheet")
' Отображаем курсор - "часы ожидания":
   DoCmd.Hourglass True
' Добавляем строку заголовков из таблицы:
   intRow = 1
   For intColumn = 1 To tdfCustomers.Fields.Count
    objXLSheet.Worksheets(1).Cells(intRow, intColumn).Value _
              = tdfCustomers.Fields(intColumn - 1).Name
   Next intColumn
' Выделяем первую запись:
   intRow = 2
   rstCustomers.MoveFirst
   Do Until rstCustomers.EOF
      For intColumn = 1 To rstCustomers.Fields.Count
         objXLSheet.Worksheets(1). _
              Cells(intRow, intColumn).Value _
               = rstCustomers.Fields(intColumn - 1).Value
      Next intColumn
' Выделяем следующую запись:
      rstCustomers.MoveNext
' и увеличиваем счетчик номера строки:
      intRow = intRow + 1
   Loop
' Проверяем, существует ли уже OfficeSolutionsRpt.xls:
   If Dir$(strSavedReport) <> "" Then
     Kill strSavedReport
   End If
' Устанавливаем для столбцов подходящую ширину:
   objXLSheet.Worksheets(1). _
        Cells(1, 1).CurrentRegion.EntireColumn.AutoFit
' и сохраняем лист:
   strMsg = "Сохраняем объект Excel как файл "  _
          & strSavedReport & "."
   MsgBox  prompt:=strMsg, _
          Buttons:=vbInformation, Title:= "Сохранение"
   objXLSheet.SaveAs strSavedReport
' Выключаем "часы ожидания".
ExitMakeXLObject:
   DoCmd.Hourglass False
```



```
' Выходим из Excel и очищаем переменную объекта:
   objXLSheet.Application.Quit
   Set objXLSheet = Nothing
' Закрываем Access:
   dbSolution.Close
   Set dbSolution = Nothing
   ws.Close
   Set ws = Nothing
   If (FindWindowByClass("Omain", 0&) <> 0) Then
      GetObject(, "Access.Application").Quit
   End If
   Exit Sub
ErrorMakeXLObject:
   If Err <> 0 Then
       strMsg = "Произошла непредвиденная ошибка. "
       strMsg = strMsg & "Ошибка = " & Err
       MsgBox  prompt:=strMsg, Buttons:=vbExclamation, _
               Title:= "Ошибка!"
   End If
   Resume ExitMakeXLObject
End Sub
```

В программе выполняется проверка существования файла OfficeSolutionsRpt.xls, если файл уже существует, его приходится удалять, чтобы предотвратить появление сообщения-запроса о перезаписи существующего файла. Обычный метод, применяемый для этого в Excel (присваивания ***Application.DisplayAlerts*** = ***False***), не работает при использовании Automation. Другой путь (с помощью метода ***SendKeys***) работает, только если объект-приложение Excel видим.

Для проверки, открыто ли окно Access, используется вызов функции ***FindWindow*** Win32 API, которая должна быть описана следующим образом:

```
Declare Function FindWindowByClass Lib "user32" _
        Alias "FindWindowA" _
           (ByVal lpClassName As String, _
            ByVal lpWindowName As Long) As Long
```

Функция ***CreateObject*** обычно запускает новый экземпляр приложения, на которое производится ссылка (как это делает и оператор ***Dim*** **As** ***New***), но из этого правила есть исключение: некоторые приложения регистрируют себя в Реестре Windows как *уникальные* (*single instance*), а это означает, что одновременно не

может быть запущено больше одного экземпляра приложения (для приложений такого типа новые экземпляры создаваться не будут, операторы будут ссылаться на запущенный экземпляр приложения). В Microsoft Office уникальными приложениями являются, например, Word и PowerPoint.

11.4. Получение доступа к объекту

К созданному объекту можно получить доступ с помощью функции ***GetObject***. В качестве параметров указывается строка, содержащая полный путь и имя файла, содержащего объект, и строка, содержащая указание класса объекта:

```
Set objXLSheet = GetObject(strSavedReport, "Excel.Sheet")
```

где **strSavedReport** – строка, содержащая путь и имя файла.

Приложение объекта можно сделать видимым для работы с ним:

```
objXLSheet.Application.Visible = True
```

В приведенной ниже процедуре открывается предварительно созданный рабочий лист, для работы с ним делается видимым приложение и сам лист (его окно), после чего на экран в окне «родительского» приложения (например, Word), из которого запускается на выполнение данная процедура, выдается сообщение. Пользователь может переключиться через панель задач в окно родительского приложения или в Excel и отредактировать лист. В родительском приложении, из которого запускается данный макрос, пользователь должен щелкнуть кнопку **OK** в диалоговом окне для продолжения работы. Если пользователь сам не завершил работу Excel, после соответствующей проверки в процедуре приложение Excel завершается без сохранения изменений, внесенных в данные (окно предупреждения на экран не выводится).

```
Sub GetObj()
   Dim strSavedReport As String
   strSavedReport = "C:\Db\OfficeSolutionsRpt.xls"
   Set objXLSheet = GetObject(strSavedReport, _
                        "Excel.Sheet")
   objXLSheet.Application.Visible = True
   objXLSheet.Parent.Windows(1).Visible = True
```



```
   MsgBox "Был открыт лист Excel"
   If Tasks.Exists(Name:="Microsoft Excel") Then
     Set myobject = GetObject("", "Excel.Application")
     MsgBox "Закрываем Excel без сохранения"
     objXLSheet.Application.DisplayAlerts = False
     objXLSheet.Application.Quit
   End If
End Sub
```

Для получения ссылки на существующий экземпляр объекта используется функция ***GetObject*** (Excel с помощью этой функции позволяет открывать файлы, но все остальные продукты Office этого не могут).

Для позднего связывания используются операторы, записанные по следующим правилам:

```
Dim ИмяОбъектнойПеременной As Object
Set ИмяОбъектнойПеременной = GetObject("ПолноеИмяФайла", _
                                "ИмяПриложения.ТипОбъекта")
```

Правило записи оператора для раннего связывания несколько отличается в части описания переменной (для нее указывается конкретный тип объекта):

```
Dim ИмяОбъектнойПеременной As Класс
Set ИмяОбъектнойПеременной = GetObject("ПолноеИмяФайла", _
                             "ИмяПриложения.ТипОбъекта")
```

где класс соответствует указанному при вызове функции типу объекта.

При использовании данной функции для открытия файла аргумент класса является необязательным. Но если функция используется для открытия объектов из файлов Excel, необходимо в качестве второго аргумента указать “Excel.Sheet” или “Excel.Chart” (а “Excel.Application” в этом случае использовать нельзя). Для открытия и отображения рабочей книги ее имя (но не весь путь) заключается в квадратные скобки.

```
Sub CreateWorkBook()
   If Dir("C:\DB\My_Book.xls") <> "" Then
     Kill "C:\DB\My_Book.xls"
   End If
   Set ObjExcel = CreateObject("Excel.Application")
   ObjExcel.Workbooks.Add
   ObjExcel.ActiveCell.Value = "Вписанные данные"
```

```
   ObjExcel.Workbooks(1).SaveAs "C:\DB\My_Book.xls"
   ObjExcel.Quit
   Set ObjExcel = Nothing
End Sub

Sub GetWorkBook()
   Set ObjExcel = GetObject("C:\DB\[My_Book.xls]", _
                            "Excel.Sheet")
   ObjExcel.Application.Visible = True
   MsgBox Prompt:="Нажмите кнопку для закрытия Excel", _
          Buttons := vbInformation
   ObjExcel.Application.Quit
   Set ObjExcel = Nothing
End Sub
```

Эти процедуры следует запускать последовательно. Первая процедура создает рабочую книгу Excel. Вторая – открывает ее.

Если в программе есть объект, ссылающийся на отдельный рабочий лист или лист диаграммы Excel, то доступ к другим рабочим листам книги можно получить с помощью свойства ***Parent*** данного объекта.

Функция ***GetObject*** запускает приложение, указанное с помощью второго аргумента, и активизирует объект, находящийся в указанном файле. Если первый аргумент является пустой строкой (""), ***GetObject*** возвращает новый экземпляр объекта указанного типа. Если этот аргумент опущен, ***GetObject*** возвращает текущий активный объект указанного типа. Если ни одного объекта указанного типа не существует, возникает ошибка. Следовательно, данную функцию можно использовать, чтобы определить, открыт ли требуемый объект:

```
Sub GetWordObj()
     On Error GoTo OLE_Error
     Dim blnIsAppOpen As Boolean
     Dim objWord As Object
     Set objWord = GetObject(, "Word.Application")
     blnIsAppOpen = True
OLE_continue:
     If Not blnIsAppOpen Then
          Set objWord = CreateObject("Word.Application")
     End If
     objWord.Application.Visible = True
     objWord.Documents.Add
     MsgBox Prompt:="Нажмите кнопку для закрытия Word", _
```



```
                    Buttons:=vbInformation
     objWord.Application.Quit
   Exit Sub
OLE_Error:
     Select Case Err
     Case 429
          blnIsAppOpen = False
          Resume OLE_continue
     Case Else
          MsgBox Prompt:="Неизвестная ошибка", _
               Buttons:=vbInformation
     End Select
End Sub
```

Приведенную выше процедуру можно запустить как макрос (например, из Word).

Этот прием можно использовать для Word, Access и PowerPoint. Но в Excel этот вариант не проходит: Excel открывает еще один объект-приложение вне зависимости от того, существовал ли он уже.

Более подробную информацию об описанных функциях можно получить в справочной системе.

11.5. Пример вычислений в Access с помощью Excel

Ниже приведен код процедуры, производящий вычисления с помощью Excel для данных, хранящихся в базе данных Access. Эта процедура запускается из Access, она содержится в базе данных OLE.MDB, в программном модуле. В процедуре осуществляется вычисление переменной амортизации для имущества, записанного в первой записи таблицы «Имущество» (Assets) базы данных OLE.MDB. Automation используется для создания рабочего листа Excel и применения для вычисления встроенной функции Excel VDB (функции ПДОБ, которая возвращает значение амортизации имущества за данный период, включая конкретные периоды, используя метод двойного процента со снижающегося остатка или иной явно указанный метод).

```
Sub ComputeVDB()
   On Error GoTo ComputeVDBError
   Dim intXLFlag As Integer
   Dim objXLSheet As Object
```

```
Dim dbSolution As Database
Dim rstAsset As Recordset
Dim intNumFields As Integer
Dim intCount As Integer
Dim intOpen As Integer
Dim strMsg As String
   Set dbSolution = DBEngine(0)(0)
   Set rstAsset = dbSolution.OpenRecordset("Assets")
   intOpen = IsAppOpen("XLMAIN")
   If intOpen = -1 Then
' Вызов функции Win32 API для регистрации Excel
' в OLE RunningObjectTable (это необходимо,
' если вы хотите использовать текущий запущенный
' экземпляр Excel, и не нанести ущерба):
      RegisterExcel hwndExcel, WM_USER + 18, 0, 0
' Получаем ссылку на открытый экземпляр Excel:
      Set objXLSheet = _
                  GetObject(, "Excel.Application")
   ElseIf intOpen = 1 Then
' Создаем новый экземпляр Excel:
      Set objXLSheet = _
                  CreateObject("Excel.Application")
' и устанавливаем флаг, чтобы закрыть Excel в конце:
      intXLFlag = 1
   Else
         strMsg = "Произошла ошибка в ComputeVDB. "
         strMsg = strMsg & "Код ошибки = " & Err
         MsgBox strMsg, vbExclamation, APP_NAME
         Exit Sub
   End If
' Отображаем  песочные "часы ожидания":
   DoCmd.Hourglass True
' Создаем рабочую книгу Excel:
   objXLSheet.Workbooks.Add
' Читаем значения из таблицы Assets
' в рабочий лист Excel:
   rstAsset.MoveFirst
   For intCount = 2 To rstAsset.Fields.Count - 1
      objXLSheet.Worksheets(1).Cells(intCount-1, 1). _
                Value = Val(rstAsset.Fields(intCount))
   Next intCount
' Устанавливаем год для которого вычисляем амортизацию:
   objXLSheet.Worksheets(1).Cells(1, 2).Value = 3
' Вычисляем амортизацию и отображаем ее:
   objXLSheet.Worksheets(1).Cells(3, 2).Formula _
               = "=VDB(A1, A2, A3, B1-1, B1, A4)"
   strMsg = "Трехлетняя амортизация по первому " _
```



```
            & "наименованию, записанному в таблице "
      MsgBox prompt:=strMsg, Buttons:=vbInformation, _
            Title:="Вычисление VDB"
   ' Если Excel не был открыт до выполнения процедуры,
   ' то выход из Excel, и очистка переменой:
      If intXLFlag = 1 Then
         objXLSheet.Workbooks(1).SaveAs "C:\DB\Res.xls"
         objXLSheet.Application.Quit
         Set objXLSheet = Nothing
      End If
   DepreciationExit:
   ' Выключаем "песочные часы ожидания":
      DoCmd.Hourglass False
      Exit Sub
   ComputeVDBError:
      strMsg = "Произошла ошибка с кодом = " & Err
      MsgBox prompt:=strMsg, Buttons:=vbExclamation, _
            Title:= "Ошибка!"
      Resume DepreciationExit
   End Sub
```

Здесь функция IsAppOpen проверяет, открыто ли приложение через поиск его окна:

```
Function IsAppOpen(strClassName As String) As Integer
   On Error Resume Next
' В случае ошибки возвращается 0
   hwndExcel = FindWindowByClass(strClassName, 0&)
 If hwndExcel <> 0 Then
   IsAppOpen = -1 ' Приложение открыто
 Else
   IsAppOpen = 1 ' Приложение еще не открыто
 End If
End Function
```

Функция Win32 API FindWindow была описана выше.

Объявление функции, используемой для регистрации Excel в OLE RunningObjectTable, выглядит следующим образом:

```
Declare Function RegisterExcel Lib "user32" _
       Alias "SendMessageA" (ByVal hWnd As Long, _
       ByVal lngMsg As Long, ByVal wParam As Long, _
       ByVal lParam As Long) As Long
```

Константа сообщения, требуемого при регистрации Excel в OLE RunningObjectTable, объявляется так:

```
Public Const WM_USER = 1024
```

При вызове функции используется также переменная – дескриптор (***Handle***) окна Excel в Windows. Эта переменная описывается так:

```
Dim hwndExcel As Long
```

эта переменная описана на уровне модуля, в котором содержатся представленные выше процедура ComputeVDB и функция IsAppOpen.

Проверить, активно ли приложение, можно и с помощью метода ***Exists*** объектов семейства ***Tasks***. Это семейство представляет все задачи, выполняющиеся в системе. Более полную информацию можно получить через просмотр объектов в Редакторе Visual Basic. Для запуска программ и добавления их в семейство объектов ***Tasks*** используется функция ***Shell***. Детальную информацию о семействе можно получить в справочной системе.

11.6. Управление связанными и внедренными объектами с помощью программного кода

Для работы со связанными и внедренными объектами можно использовать программный код, с помощью которого можно устанавливать ссылки на материалы, созданные в другом приложении, заблокировать ссылки, отредактировать или обновить ссылки, чтобы они указывали на другие источники. Кроме того, можно создавать внедренные объекты, редактировать их.

В приложении-клиенте можно активизировать внедренный объект с помощью семейства ***OLEObjects***.

Управление всеми связанными и внедренными объектами (рисунками, диаграммами, таблицами и т.п.) осуществляется сходными средствами. Приведенный ниже программный код демонстрирует возможности по созданию таких объектов и управлению ими.

```
Sub InsertLinkedObject();
   Const Path = "ПолноеИмяГрафическогоФайла"
   Dim OLEObj As OLEObject
   Set OLEObj = ActiveSheet.OLEObjects.Add _
              (FileName:=Path, Link:=True)
   OLEObj.Name = "Picture1"
   I = OLEObj.Index
   MsgBox "На активный лист вставлен рисунок. " _
        & "Его индекс=" & Trim(Str(I)) _
        & ". Его имя - " & OLEObj.Name
End Sub
```


При выполнении данного макроса в Excel на активном рабочем листе размещается рисунок и происходит связывание данного объекта с исходным графическим файлом, который создан в графическом редакторе (например, Paint). При редактировании данного рисунка в исходном файле изменяется и внедренный объект (при обновлении связи).

Для внедрения объекта на рабочий лист (без установки связи с исходным файлом) достаточно указать значение аргумента ***Link*** равным ***False***.

Для редактирования объекта, вставленного на рабочий лист, можно выполнить метод **Activate** для этого объекта:

```
Объект.Activate
```

в качестве ссылки на объект может быть указана объектная переменная или объект может быть выбран из семейства объектов на рабочем листе по индексу (по имени):

```
РабочийЛист.OLEObjects(Индекс)
```

Аналогично осуществляется работа с другими объектами, располагаемыми на рабочем листе.

Вставить рисунок из файла на рабочий лист можно также с помощью оператора вида:

```
ActiveSheet.Pictures.Insert("ПолноеИмяГрафическогоФайла")
```

Для редактирования такого рисунка его можно выделить с помощью оператора

```
ActiveSheet.Shapes(Индекс).Select
```

и внести изменения с помощью панели инструментов «Настройка изображения».

Средства управления различными объектами можно более подробно изучить с помощью справочной системы. Найти нужные методы можно также, записав макрос с интересующей последовательностью операций.

Для размещения рисунка в документе Word можно было бы использовать следующий оператор:

```
Set OLEObj = ActiveDocument.Shapes.AddPicture _
     (Anchor:=Selection.Range, _
     FileName:="ПолноеИмяГрафическогоФайла", _
     LinkToFile:=True, SaveWithDocument:=True)
```

или оператор

```
Set OLEObj = ActiveDocument.Shapes.AddOLEObject _
     (Anchor:=Selection.Range, _
     ClassType:="Paint.Picture", _
     FileName:=" ПолноеИмяГрафическогоФайла ", _
     LinkToFile:=True, DisplayAsIcon:=False)
```

где переменная OLEObj может иметь тип ***Object***. Эту переменную можно использовать для выполнения последующих действий над вставленным в документ объектом.

Внедрить рабочий лист Excel в документ Word можно с помощью оператора вида

```
Set OLEObj = ActiveDocument.Shapes.AddOLEObject _
             (Anchor:=Selection.Range, _
             ClassType:="Excel.Sheet", _
             FileName:="", LinkToFile:=False, _
             DisplayAsIcon:=False)
```

связь с файлом в данном случае не устанавливается, рабочий лист создается «на месте».

Внедренный рабочий лист можно отредактировать «на месте», выполнив следующий оператор:

```
ActiveDocument.Shapes(Индекс).OLEFormat.DoVerb _
                    VerbIndex:=wdOLEVerbPrimary
```

или с помощью оператора

```
ActiveDocument.Shapes(Индекс).OLEFormat.DoVerb _
                     VerbIndex:=wdOLEVerbOpen
```

который запускает Excel для работы с внедренным в активный документ Word рабочим листом. В качестве индекса может использоваться номер или имя объекта.

Доступ к объекту можно получить через объектную переменную, которой предварительно нужно присвоить значение с помощью оператора ***Set*** при создании объекта (как это было сделано в приведенных выше примерах) или при его выборе:

```
Set OLEObj = ActiveDocument.Shapes(Индекс)
```

Более подробно синтаксис использованных в примерах операторов, назначение параметров и их возможные значения можно изучить, воспользовавшись информацией, предоставляемой справочной системой.

В приложении Microsoft Office можно использовать код

другого приложения. Такая необходимость возникает, когда в документ приложения включен объект, созданный другим приложением. Тогда для выполнения операций над данным объектом в контроллере Automation, управляющем составным документом, в который включен объект, необходимо обратиться за услугами к серверу Automation, обслуживающему этот объект.

Для исполнения кода сервера Automation в процедуре контроллера Automation необходимо:

Загрузить нужную библиотеку объектов сервера, выполнив в меню **Сервис** редактора VBA команду **Ссылки**.

Добавить код, проверяющий наличие уже запущенного экземпляра сервера Automation.

Написать код, использующий Automation, за кодом получения ссылки на объект Automation.

При написании кода следует учитывать, что есть инструкции и приемы, поддерживаемые только сервером, которые нельзя исполнить в контроллере.

Код сервера Automation можно перенести в процедуру контроллера через буфер. А для сервера код можно сгенерировать, воспользовавшись макрорекодером.

Вопросы для самопроверки

1. Дайте определение технологии ActiveX.
2. Приведите понятие компонентной модели объектов COM. Каким образом COM связана с OLE Automation?
3. Дайте определение клиента и сервера. Какие приложения MS Office могут выступать в качестве серверов и в качестве клиентов?
4. Какие модели интегрированных приложений вы знаете? Дайте пояснение.
5. Каким образом можно создать объект Automation? Приведите примеры.
6. В чем отличия раннего и позднего связывания? Приведите примеры.
7. Каким образом можно получить доступ к объектам Automation? Приведите примеры.
8. Приведите примеры управления объектами с помощью программного кода.

Глава 12. Организация доступа к внешним данным

Компоненты Microsoft Office являются хорошей основой для создания клиентских приложений. Например, можно использовать Word при построении клиентской части приложения, если при создании отчетов или публикаций в них включаются данные из внешних источников. При необходимости статистического анализа данных для разработки приложения лучше использовать Excel.

При разработке приложений на базе Microsoft Office возникает проблема получения клиентским приложением необходимой информации из внешних баз данных. Далее рассматриваются две технологии доступа к данным (ODBC и DAO), реализованные в продуктах Microsoft.

12.1. Доступ к данным из клиентских приложений

Существует четыре основных подхода к реализации доступа к данным из продуктов Microsoft Office:

- использование шаблонов Word и механизма слияния для хранения и изменения запросов;
- использование ODBC API (Open Database Connectivity – открытый интерфейс доступа к базам данных);
- использование DAO (Date Access Objects – объекты доступа к данным);
- использование технологии RDO (Remote Date Objects – объекты удаленного доступа к данным) для доступа к источникам данных SQL в рамках локальной или корпоративной сети.

В дополнение к возможностям, предоставляемым разработчикам DAO и ODBC, созданы надстройки для работы с данными в приложениях Microsoft Office. Например, в Access имеются встроенные средства для импорта данных и связи с Excel; пользователь Excel имеет возможность создавать экранные формы Access, получить доступ к командам Access.

В состав Microsoft Office входит приложение Microsoft Query (упрощенная версия средства создания запросов Microsoft Access), с помощью которого пользователи Word и Excel могут получать информацию из источников данных ODBC.

Далее кратко рассматриваются средства организации доступа к внешним данным.

12.2. Интерфейс ODBC

Для обеспечения свободного доступа к данным, хранящимся в различных форматах, необходим некоторый стандартный способ соединения с базами данных и стандартный язык, обеспечивающий доступ к данным и манипулирование ими. ODBC обеспечивает соединение с источниками данных. Ввод и получение данных осуществляется с помощью структурированного языка запросов SQL (Structured Query Language). ODBC использует конструкции языка SQL для связи с различными системами управления базами данных (СУБД) независимо от их разработчиков и обеспечивает следующие функции работы с базами данных (БД):

- создание, изменение и удаление таблиц;
- создание и удаление представлений и индексов;
- отбор, обновление, вставка и удаление полей таблиц.

Реляционная БД является набором реляционных таблиц. *Таблицы* являются хранилищем данных, а *представления* данных (VIEW) предоставляют пользователю возможность увидеть выборки из этих таблиц. Представления можно использовать для просмотра системных таблиц, которые содержат основные сведения о БД. Представления данных являются частью схемы БД и могут восприниматься как виртуальные таблицы. В реляционной БД необходимо иметь возможность обновления и модификации представления данных.

Помимо основных функций для работы с данными, обеспечиваемых диспетчером драйверов, многие СУБД предоставляют пользователю дополнительные средства и функции. Для реализации этих дополнительных возможностей клиентские приложения используют драйверы ODBC конкретных СУБД, поставляемых обычно с СУБД.

ODBC представляет собой библиотеку функций, которая позволяет прикладной программе обращаться к данным различных СУБД, используя язык SQL.

Архитектура ODBC включает четыре основных компонента: прикладная программа (клиентское приложение), Менеджер (диспетчер) драйверов, драйвер(ы) и источник или источники данных.

Приложения, использующие ODBC, не обращаются к источникам данных напрямую, они работают через Диспетчер драйверов, применяя SQL.

Менеджер драйверов является библиотекой DLL, которая загружает драйверы, обеспечивая единственную точку входа в функции ODBC для различных драйверов. Основное его назначение состоит в загрузке драйверов, которые используют вызовы функций ODBC. Драйверы ODBC осуществляют непосредственный доступ к источникам данных. Драйверы принимают вызовы функций и взаимодействуют с источниками данных. Для реализации своих функций драйвер может управлять коммуникационными протоколами между данной прикладной программой и источником данных, когда программа осуществляет вызов функции подсоединения к источнику данных. После установки соединения драйвер способен управлять запросами к СУБД, которые выполняются прикладной программой, обеспечивая необходимую передачу информации и возврат результатов.

Схема взаимодействия компонентов ODBC представлена на рис. 12.1.

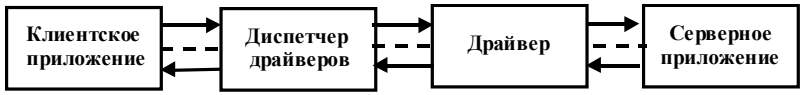


Рис. 12.1. **Архитектура ODBC**

На рис. 12.2 представлен основной алгоритм использования ODBC в прикладных программах.

Основные функции ODBC подразделяются на несколько групп:

– назначение и отмена назначения идентификатора окружения, идентификатора соединения и идентификатора оператора;

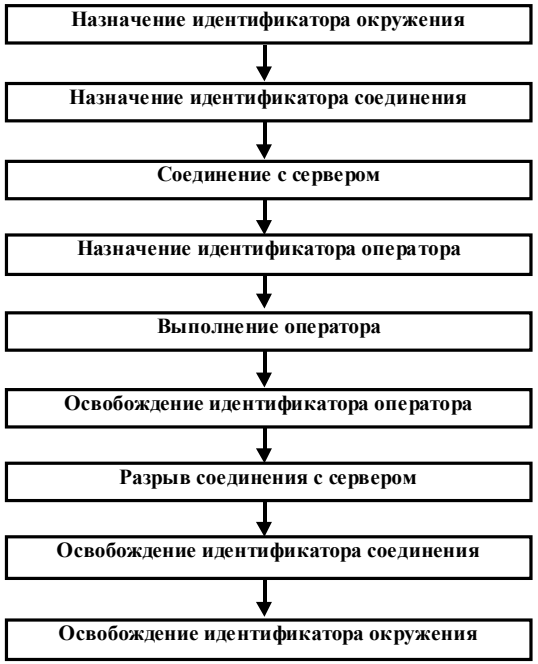


Рис. 12.2. **Схема использования ODBC**

– соединение;
– выполнение SQL-операторов;
– получение результатов;
– управление транзакциями;
– идентификация ошибок и смешанные функции.

Идентификатор окружения определяет базу данных, он указывает на область памяти для глобальной информации (содержит сведения обо всех соединениях с базами данных и информацию о том, какое соединение является текущим).

Идентификатор соединения определяет соединение с БД, указывает на область памяти с информацией о конкретном соединении. Прикладная программа может устанавливать соединения одновременно к нескольким серверам – источникам данных, а также несколько различных соединений к одному и тому же серверу. Но в любой момент времени только одно соединение является текущим.

Идентификатор оператора определяет отдельный SQL-оператор. Он создается до выдачи программой SQL-запроса.

Выполнение функции назначения ведет к выделению памяти и определяет требуемые структуры данных, присваивает значение идентификатору, которое используется для доступа к этим структурам данных. Функции отмены назначения освобождают идентификаторы и соответствующие им области памяти.

Функции управления транзакциями позволяют завершить транзакцию или выполнить откат на ее начало.

Установить диспетчер ODBC и набор драйверов можно в процессе установки Microsoft Office. После установки значок ODBC появляется в Панели управления Windows. Диспетчер драйверов используется для настройки источников данных ODBC. Чтобы создать новый источник данных, нужно выбрать драйвер нужной СУБД в Диспетчере драйверов, настроить его для работы с конкретной БД и присвоить источнику уникальное имя (DSN – Data Source Name).

Как уже было сказано, ODBC обеспечивает доступ к данным, независимый от поставщиков. Например, при вставке информации из базы данных Microsoft Access в документ Word через буфер или присоединении ее слиянием с источником данных с использованием ODBC, ODBC читает непосредственно файл данных, при этом не требуется открывать Microsoft Access. Однако чтобы использовать запрос Microsoft Access как источник данных, необходимо открыть Microsoft Access.

Если нужно использовать запрос, следует использовать DDE вместо ODBC (так как DDE автоматически открывает Microsoft Access при чтении файла данных). Для этого сначала, как обычно, нужно вставить информацию из базы данных или присоединить ее слиянием с источником данных. В диалоговом окне «Открытие источника данных» устанавливается флажок «Выбрать метод» перед тем, как нажать кнопку **Открыть**. Затем, каждый раз при вставке информации из базы данных или присоединении ее слиянием с источником данных, Word открывает диалоговое окно «Подтверждение источника данных», в котором можно выбрать команду DDE или другой параметр преобразования.

Если возможности ODBC не устраивают разработчика, он может использовать API конкретной СУБД или DAO.

12.3. Использование DAO для доступа к данным

Объекты доступа к данным (DAO) создавались как объектно-ориентированный интерфейс для ядра баз данных Jet фирмы Microsoft. Ядро Jet обеспечивает возможность доступа к данным независимо от особенностей интерфейса конкретной системы управления базами данных. После того как Jet стал сервером Automation, стало возможным использовать DAO для доступа к данным из любого клиента Automation (Excel, Word и PowerPoint).

Версия DAO, включенная в Office, позволяет манипулировать данными в обход ядра Jet – связываться напрямую с источниками данных ODBC можно через ODBCDirect (рис. 12.3). Прямой доступ к источникам позволяет повысить производительность и сэкономить ресурсы.

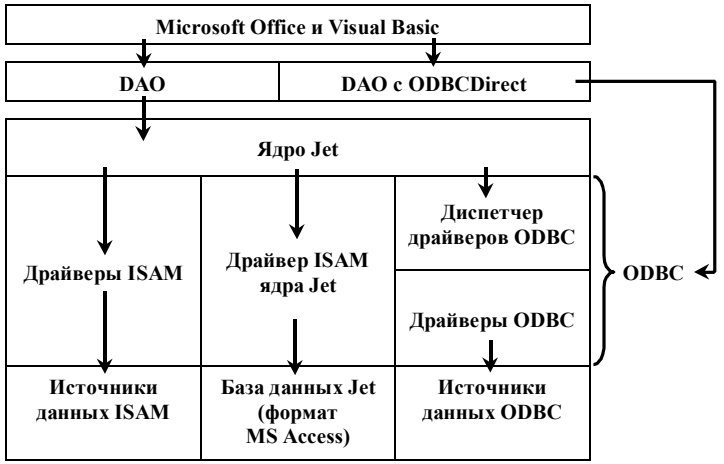


***Рис. 12.3*. Доступ к данным из MS Office и Visual Basic**

DAO – это унифицированный набор объектов для доступа к данным. То есть с его помощью пользователь получает стандартный объектно-ориентированный интерфейс доступа к различным типам данных, начиная от баз данных Access, до баз данных ISAM (Indexed Sequential Access Method – индексно-последовательный метод доступа к данным, используемый в БД для персональных компьютеров) и SQL.

Доступ к DAO из VBA осуществляется через OLE Automation, но использовать DAO можно лишь после установки в Visual Basic Editor ссылки на объектную библиотеку Microsoft DAO 3.5. С помощью DAO можно осуществлять запросы на выборку и изменение данных, выполнять операции по просмотру и поиску данных, создавать базы данных, определения таблиц, спецификации запросов и зависимостей между таблицами, присоединяться к удаленным источникам данных.

12.3.1. Модель объектов DAO

Как видно из рис. 12.3, DAO включает две объектные модели. Тип используемой модели (рис. 12.4, 12.5) зависит от того, используется ли ODBCDirect.

Операции, выполняемые с помощью DAO, можно разделить на три категории:

- *запросы* – операции импорта и экспорта данных, включающие выборку, добавление, редактирование и удаление записей;
- *операции изменения структуры БД* – операции создания, изменения и удаления БД, спецификаций таблиц и запросов, полей, индексов и связей (на выполнение этих операций могут существовать ограничения в зависимовти от типа используемой внешней базы);
- *операции, обеспечивающие защиту данных в многопользовательской среде,* – создание сеансов под различными пользовательскими именами, установка и идентификация пользователей и групп.

Объекты, включенные в иерархическую объектную модель DAO, используются для реализации этих операций. В модель включены два типа объектов: постоянные и временные. *Постоянные* объекты могут быть сохранены в MDB-файле или в системном файле БД. *Временные* объекты не сохраняются в файлах, любые свойства временного объекта являются временными.

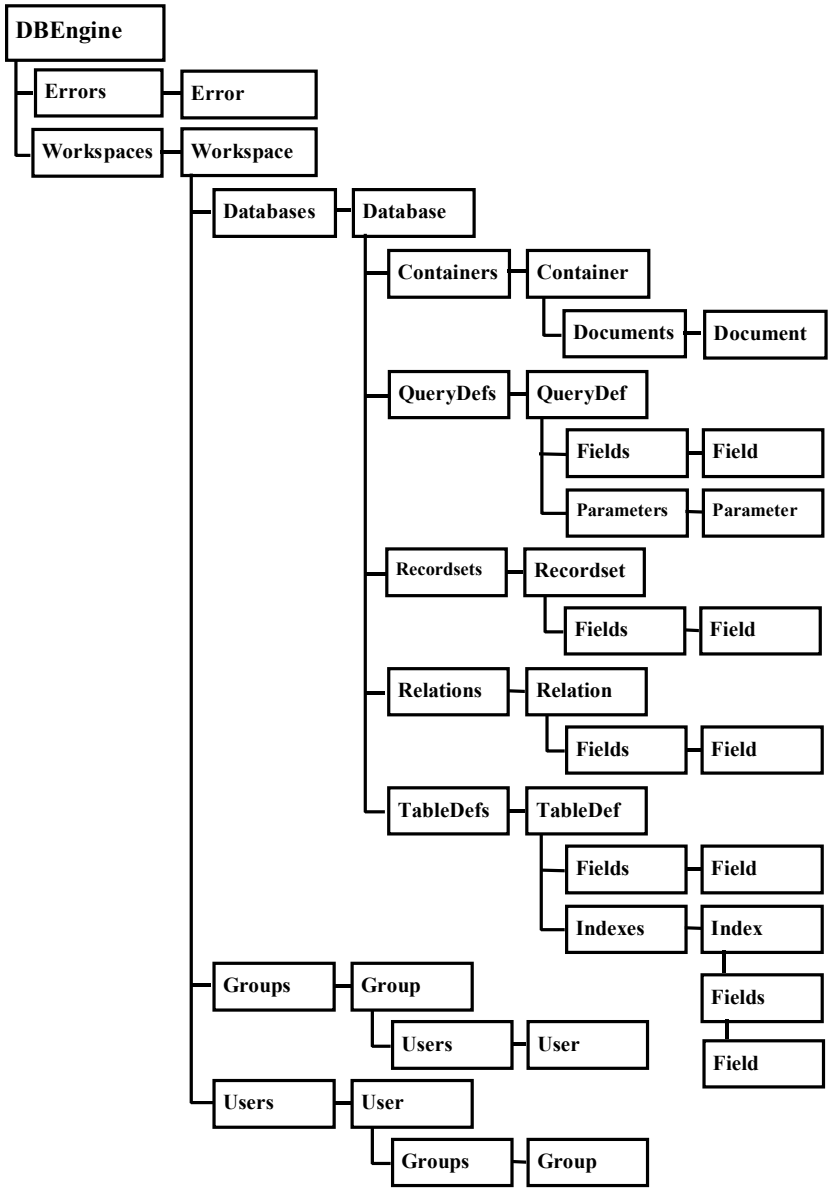


Рис. 12.4. **Модель объектов доступа к данным для рабочей области Microsoft Jet**

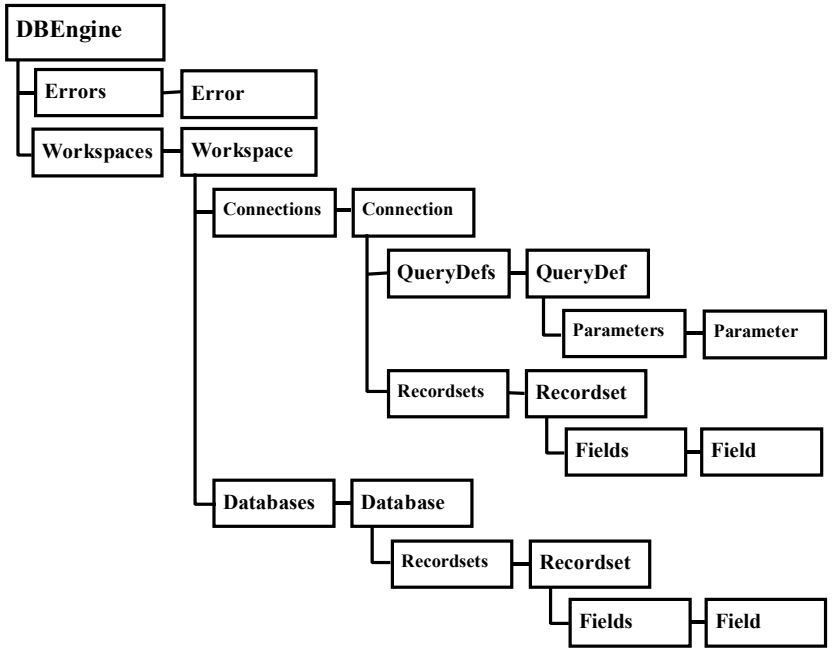


Рис. 12.5. **Модель объектов доступа к данным для рабочей области ODBCDirect**

При доступе в приложении к объекту DAO он либо создается, либо ссылка на него задается через семейство, содержащее его. На временный объект можно ссылаться через семейство только после того, как он создан. Постоянные объекты можно создавать подобно временным. После создания постоянного объекта он в большинстве случаев не сразу становится постоянным, а только после добавления его к своему семейству. Объекты, не являющиеся постоянными, могут использоваться для доступа к постоянным объектам, представляющим сохраненные базы данных, таблицы, запросы и зависимости.

Независимо от того, какая модель используется, перед началом работы с DAO необходимо создать объект ***Workspace*** (рабочая область или сеанс работы с БД), в котором будут производиться все операции с БД.

12.3.2. Создание рабочей области и открытие источника данных

При доступе к данным с помощью ядра Microsoft Jet используется рабочая область Microsoft Jet, для создания которой задается зарезервированная константа ***dbUseJet***. В случае доступа к источнику данным ODBC используется рабочая область ODBCDirect, для создания которой используется зарезервированная константа ***dbUseODBC***.

Метод ***CreateWorkspace*** определяет название рабочей области (первый параметр), второй и третий параметры задают имя и пароль, необходимые для доступа к данным, последний параметр определяет тип создаваемой рабочей области. В качестве результата возвращается ссылка на рабочую область.

Следующий программный код создает рабочую область Microsoft Jet:

```
Dim wsJet As Workspace
Set wsJet = DBEngine.CreateWorkspace("MyWorkspace", _
                    "Admin", "", dbUseJet)
```

А приведенные ниже операторы создают рабочую область ODBCDirect:

```
Dim wsODBC As Workspace
Set wsODBC = CreateWorkspace("MyWorkspace", _
                    "Admin", "", dbUseODBC)
```

(объект ***DBEngine*** используется по умолчанию).

После создания рабочей области можно открыть источник данных (т.е. соединиться с источником данных). Порядок открытия источника данных зависит от типа рабочей области.

Если создана рабочая область Microsoft Jet, то для соединения с источником используется метод ***OpenDatabase***, в качестве параметра которого указывается полный путь к базе данных:

```
Dim dbObj As Database
Set dbObj = _
    wsJet.OpenDatabase("C:\MyDataBases\DAO.mdb")
```

При использовании рабочей области ODBCDirect для открытия источника данных применяется метод ***OpenConnection***, который имеет один обязательный аргумент – строку, задающую имя источника данных. В начале этой строки необходимо указать тип источника данных (ODBC), вся строка является значе-

нием свойства ***Name*** объекта ***Connection***. Перед использованием ичточника необходимо убедиться в том, что в Диспетчере драйверов ODBC описан соответствующий источник данных (в случае необходимости нужно создать новый источник данных). В следующем примере устанавливается соединение с источником данных, имеющим имя «Publishers»:

```
Dim dsObj As Connection
Set dsObj = _
        wsODBC.OpenConnection("ODBC;DSN=Publishers")
```

Регистрация источника данных ODBC осуществляется с помощью Диспетчера драйверов ODBC, доступ к которому можно получить через Панель управления Windows. Источник данных ODBC можно зарегистрировать программным путем, используя ODBC API.

После соединения с источником данных можно производить операции по обработке данных.

12.3.3. Исследование структуры таблиц БД

Для исследования структуры базы данных можно воспользоваться свойствами семейства ***TableDefs***. Свойство ***Count*** возвращает количество таблиц. Свойство ***Name*** объекта из семейства ***TableDefs*** возвращает имя таблицы в БД. Для обращения к конкретной таблице можно использовать ее индекс (можно указать номер или имя таблицы).

Следующий пример позволяет вывести на экран количество таблиц в БД и их имена:

```
Dim tdf As TableDef
Dim strMsg As String
   . . . . . .
strMsg = "В данной БД " & dbObj.TableDefs.Count
strMsg = strMsg & " таблиц с именами:" & vbCrLf
For Each tdf In dbObj.TableDefs
   strMsg = strMsg & tdf.Name & vbCrLf
Next
MsgBox Prompt:=strMsg, Buttons:=vbInformation, _
                  Title:="Структура БД"
```

Далее можно выяснить структуру каждой таблицы. Для этого можно воспользоваться семейством ***Fields***. Свойства этого семейства позволяют определить количество полей, их имена и т.д.

12.3.4. Операции над данными

Для непосредственной работы с данными используется объект ***Recordset***. Существует несколько типов объектов ***Recordset*** (типы обозначаются константами):

dbOpenSnapshot	**Статическая (неизменяемая) копия записей.**
dbOpenDynaset	**Набор записей, полученных в результате выполнения запроса, который можно изменять произвольным образом:** редактировать, добавлять и удалять данные.
dbOpenTable	**Набор записей, представляющий все записи одной таблицы.** Набор записей этого типа можно изменять произвольным образом. Данный тип используется по умолчанию при работе с Microsoft Jet, если указать таблицу как источник записей для объекта ***Recordset***.
dbOpenDynamic	**Обновляемый объект *Recordset*.** Используется только в рабочей области ODBCDirect.
dbOpenForwardOnly	**Неизменяемый объект *Recordset*.** Используется по умолчанию в рабочей области ODBCDirect.

Для получения данных используются операторы SQL, которые передаются в качестве строковых параметров методу ***OpenRecordset***. Например:

```
Dim strSQL As String
Dim rsObj As Recordset
. . . . . .
srtSQL = "SELECT * FROM Products WHERE Units > 100;"
Set rsObj = dbObj.OpenRecordset(strSQL, dbOpenDynaset)
```

Если источником данных является таблица, то по умолчанию для объекта ***Recordset*** используется табличный тип. Статический набор записей используется, когда данные нужны только для просмотра (без редактирования). Если при работе с данными записи изменяются, то следует использовать динамические наборы.

Для перемещения к последней записи в наборе используется метод ***MoveLast***. Свойство ***RecordCount*** дает возможность получить количество записей (после перемещения к последней записи). Для перемещения к предыдущей/следующей записи в таб-

лице используются методы ***MovePrevious*** и ***MoveNext***. Для поиска можно использовать метод ***FindFirst***.

Процесс *редактирования данных* состоит из нескольких этапов: создание (открытие) изменяемого объекта ***Recordset***; выбор записи, подлежащей редактированию (для определения записи, удовлетворяющей заданным критериям, в таблице следует воспользоваться методом ***Seek*** в сочетании с табличным индексом, можно использовать и методы перемещения к записям в наборе); использование метода ***Edit*** объекта ***Recordset*** для помещения записи в буфер для редактирования; изменение значений полей записи; вызов метода ***Update*** для сохранения обновленной записи.

Для *добавления записи* нужно выполнить несколько шагов: создать (открыть) изменяемый объект ***Recordset***; вызвать метод ***AddNew***; установить значения всех обязательных полей; вызвать метод ***Update*** для сохранения изменений в объекте ***Recordset***.

Операции изменения данных могут быть выполнены как единая транзакция, которая может быть инициализирована и завершена (либо может быть выполнен ее откат) с помощью соответствующих методов. Для инициирования процесса изменения данных следует вызвать метод ***BeginTrans*** объекта ***Workspace***. Вызов метода ***CommitTrans*** объекта ***Workspace*** для ***Recordset*** сохраняет изменения. В обработчик ошибок, которые могут возникнуть при добавлении/изменении записей, можно вставить вызов метода ***RollBack*** объекта ***Workspace*** для восстановления исходного состояния объекта ***Recordset*** (отката транзакции).

Для *удаления записи* следует открыть объект ***Recordset***; выбрать удаляемую запись; вызвать метод ***Delete***.

Для копирования записей (например, в массив) можно воспользоваться методом ***GetRows*** объекта ***Recordset***.

В диапазон рабочего листа Excel можно скопировать данные объекта ***Recordset*** непосредственно с помощью метода ***CopyFromRecordset*** объекта ***Range***.

При завершении работы все объекты типа ***Recordset***, ***Database*** и ***Workspace*** закрываются с помощью метода ***Close***. Объекты освобождаются присваиванием объектным переменным значения ***Nothing*** с помощью оператора ***Set***.

12.4. Интеграция данных и работа в сети

Большая часть электронной информации хранится в виде электронных таблиц и текстовых документов. При этом возникают проблемы, связанные с интеграцией данных, хранящихся в различных источниках, с необходимостью согласования работы с ними многих пользователей со своих рабочих мест.

Рассматриваемые методы доступа к даным (ODBC и DAO) позволяют облегчить работу с данными, обеспечивая возможности доступа к ним (в частности, к электронным таблицам) из различных приложений Office; создания клиентских приложений, позволяющих использовать внешние данные; автоматизации процесса интеграции данных, хранящихся в различных источниках, в централизованную базу данных, управляемую СУБД.

12.4.1. Доступ к данным Excel из СУБД Access

Современные СУБД включают мощные средства, позволяющие пользователю получить доступ к данным, находящимся во внешних по отношению к СУБД источниках (средства импорта данных, доступ к данным через ODBC и т.д.).

СУБД Access, являясь одной из наиболее мощных СУБД, в то же время предоставляет в распоряжение пользователей-непрограммистов удобные средства для получения доступа к данным, формирования отчетов и т.п. Пользователь, работающий в среде Microsoft Office, может работать с данными в привычной для себя среде (например, реализовывать вычисления, анализировать данные с помощью Excel) и использовать возможности СУБД для организации удобного ввода и поиска данных, интеграции данных и синхронизации работы с ними, формирования отчетов.

Рассмотрим пример работы с таблицами Excel в СУБД Access.

Средства, включенные в приложения MS Office, позволяют каталогизировать данные, хранящиеся в различных узлах сети. Для этого можно использовать, например, гиперссылки, создав каталог ресурсов на основе Word.

Объекты MS Office позволяют управлять обменом сообщениями по электронной почте, хранить пересылаемые по почте сообщения в базах данных.

12.5. Выбор метода доступа к данным

Выбор подходящего метода доступа к данным основывается на анализе задач, стоящих перед разработчиком.

В приложениях Excel, используемых для просмотра или анализа данных, не требующих их обновления, нужно использовать сводные таблицы, которые позволяют организовать доступ к внешним источникам данных.

В приложениях, требующих как импорта, так и экспорта данных, следует использовать DAO.

Так как поток данных в сводную таблицу однонаправлен, то для создания приложения, дающего возможность пользователю изменять данные в сводной таблице, необходимо разработать специальный интерфейс четырехэтапного процесса управления потоком данных:

С помощью ODBC данные передаются в сводную таблицу для просмотра и анализа.

Если необходимо изменить данные в сводной таблице, их следует скопировать в диапазон на рабочем листе и там модифицировать.

После внесения изменений данные экспортируются во внешнюю базу данных с использованием технологии DAO.

Сводная таблица обновляется с помощью метода ***Refresh*** объекта ***PivotTable***.

DAO предоставляет возможность доступа к данным как через ядро базы данных Jet, так и прямо через ODBC.

Ядро Jet является собственной внутренней системой программы Access, распространяемой вместе с Visual Basic и Office. Ядро БД Jet позволяет осуществлять доступ как к локальным источникам информации (БД Access), так и к согласованным с ODBC источникам данных, таким как SQL Server или Oracle. Jet обладает следующими особенностями:

– в одном запросе можно комбинировать данные из источников различных типов (например, данные из локальных

таблиц Access и таблицы DB2, находящейся на сервере);
– обеспечивается объектная модель изменения структуры БД, т.е. таблиц, индексов, полей и связей, средств защиты БД для пользователей и групп.

Эти возможности увеличивают наклалные расходы (особенно при доступе к серверным источникам данных (SQL Server, Sybase, Oracle)). Это обусловлено, в частности, самим процессом выполнения запроса: для связи с серверными БД используется ODBC, следовательно, операции проходят два уровня (Jet и ODBC). Поэтому при обращениях к серверным источникам данных предпочтительным вариантом является использование доступа через ODBCDirect, который не взаимодействует с Jet, а обращается прямо к ODBC.

Учитывая сказанное выше, можно сделать следующие выводы:

– для доступа к БД «настольных» СУБД типа Access, Dbase, Paradox или FoxPro следует использовать объекты доступа к данным DAO Jet;
– для управления структурой БД нужно использовать DAO Jet;
– для комбинирования данных из двух или более источников различного типа используется DAO Jet;
– для доступа к БД серверного типа используются объекты доступа к данным ODBCDirect.

Более полную информацию можно получить в справочной системе.

Вопросы для самопроверки

1. Какие подходы к организации доступа к данным из клиентских приложений MS Office вы знаете?
2. Какова роль приложения Microsoft Query?
3. Какие возможности доступа к данным обеспечивает ODBC?
4. Какова роль языка SQL при организации доступа к данным?
5. Опишите основные элементы архитектуры ODBC.

6. Каков общий алгоритм использования ODBC для доступа к данным?
7. Опишите основные функции ODBC. На какие группы они разбиваются?
8. Каким образом устанавливается ODBC? Каков порядок подключения к различным источникам?
9. Определите метод доступа к данным DAO.
10. Опишите общую схему доступа к данным из Microsoft Office и Visual Basic.
11. Опишите модель объектов DAO. Каковы ее особенности?
12. На какие группы разбиваются операции, выполняемые с помощью DAO?
13. Какие действия нужно выполнить для использваония источника данных? Каковы их особенности при использовании разных моделей?
14. Что такое рабочая область?
15. Какие возможности имеются для работы со структурой базы данных?
16. Какие операции над данными можно выполнять?
17. Какие типы объектов используются для выполнения операций над данными? В чем их особенность?
18. Какие методы применяются для работы с данными (перемещения по записям, поиска, просмотра и редактирования, добавления и удаления)?
19. Дайте понятие транзакции. Какие возможности по организации транзакций имеются?
20. Опишите особенности выбора метода доступа к данным. Приведите примеры. Обоснуйте выбор при решении конкретных задач.

Глава 13. Особенности раннего и позднего связывания

При создании приложений часто возникает необходимость организации взаимодействия нескольких офисных программ: например, данные хранятся и обрабатываются в электронных таблицах Excel, но для подготовки отчетов используются шаблоны документов Word. В этом случае одно из приложений становится клиентом другого. В данном случае в качестве клиента выступает Excel, который запрашивает обслуживание у Word, являющегося сервером. Технология связывания рассматривалась выше. Однако при переносе автоматизированных документов, приложений, созданных на основе MS Office, на другой компьютер могут возникнуть проблемы несовместимости версий программ. Эти проблемы рассматриваются в данной главе.

13.1. Задача связывания

Для взаимодействия клиента и сервера, то есть для того, чтобы клиент мог использовать свойства и методы сервера, серверный объект должен быть *связан* с переменной на стороне клиента. Связывание определяет, как клиент соединяется с конкретным объектом-сервером. В VBA форма связывания в первую очередь определяется тем, как разработчик приложения-клиента объявляет объектные переменные.

При работе с объектами Automation может быть использовано *раннее* и *позднее* связывание. Раннее связывание подразумевает, что во время компиляции (в случае VBA заранее, так как эта среда является интерпретирующей) заранее известно, какой интерфейс будет использован для доступа к объектной переменной. При позднем связывании до создания объекта о нем ничего не известно.

13.1.1. Раннее связывание

При раннем связывании тип объектной переменной явно указывается при ее объявлении. Например, в следующей строке объявляется переменная, которая может содержать указатель

только на объект Microsoft Excel Application:

```
Dim ExlApp As Excel.Application
```

При попытке сохранить в такой переменной ссылку на объект другого типа произойдет ошибка. Если используемый объект принадлежит другому приложению или внешнему компоненту (например, используется приложение Word из Excel), то предварительно необходимо установить ссылку на библиотеку типов используемого компонента.

Использование раннего связывания имеет два основных преимущества: повышение производительности приложения во время выполнения и возможность интеллектуальной подсказки синтаксиса обращения к свойствам и методам используемого объекта. Первое преимущество достигается за счет оптимизации обращения к объекту, а второе – за счет использования информации из библиотеки типов.

Однако ранее связывание имеет один существенный недостаток. Так как тип переменной задается явно, то переменная не может содержать ссылку ни на какой другой тип. Данный факт приводит к невозможности разработки программных проектов документов, работающих на различных версиях Microsoft Office, так как каждая из версий Office имеет свою версию библиотеки типов, хотя и при совпадении имен объектов.

Продемонстрируем изложенную проблему на примере – разработаем рабочую книгу Excel в Office 2003, в программном проекте которой добавим процедуру вызова функциональности Word соответствующей версии. Для этого установим в проекте ссылку на библиотеку объектов (рис. 13.1) и заметим, что версия библиотеки имеет версию 11.

Затем введем приведенный ниже прграммный код:

```
Public Sub TestLinking()
    Dim WordApp As New Word.Application
    Dim Doc As Word.Document
    WordApp.Visible = True
    Set Doc = WordApp.Documents.Add
    Doc.Range.Text = UCase("Text in Document.")
End Sub
```

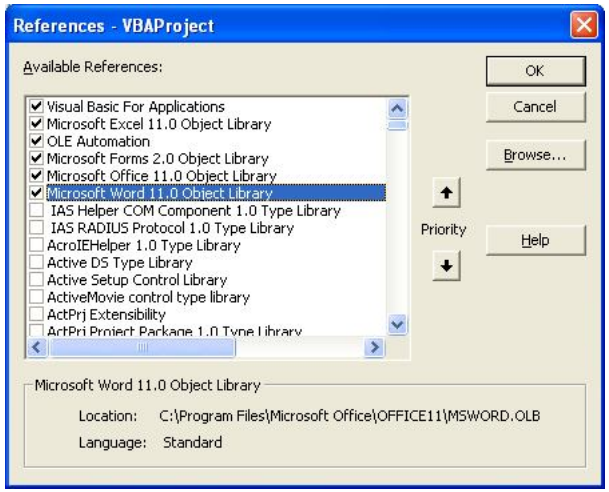



Рис. 13.1. **Диалоговое окно установки ссылок**

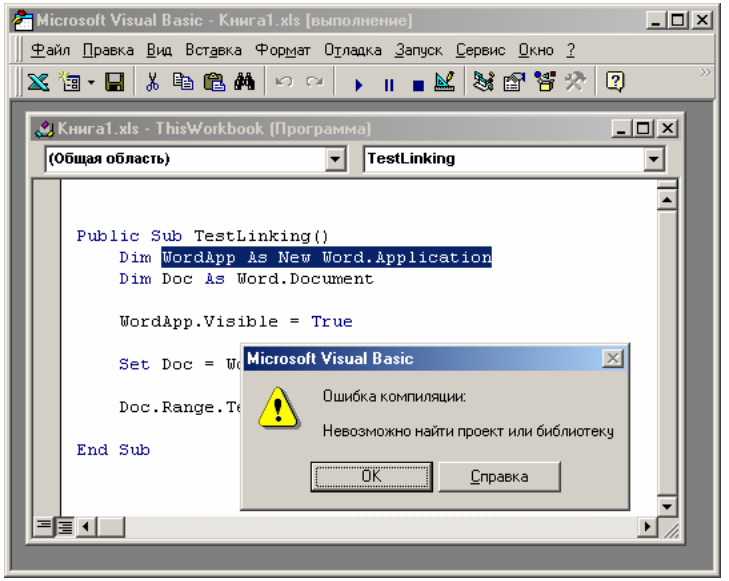


Рис. 13.2. **Сообщение об ошибке при смене версии**

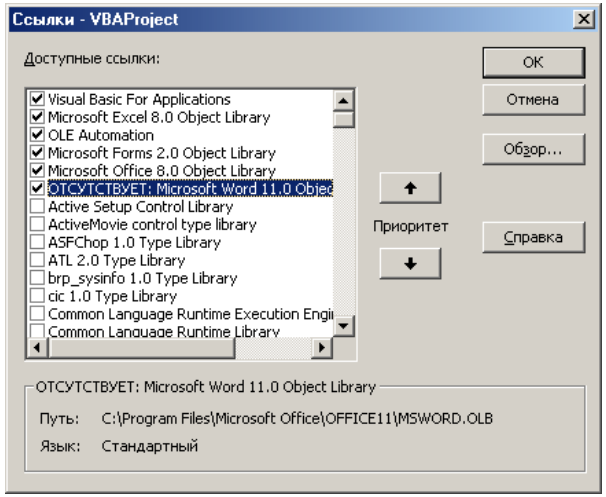


Рис. 13.3. **Настройка (изменение) ссылок на библиотеки**

При попытке выполнения макроса в Excel 97 выдается сообщение об ошибке (рис. 13.2). При просмотре ссылок проекта указывается, что библиотека Microsoft Word 11.0 Object Library не найдена (рис. 13.3). Толька при удалении ссылки на отсутствующую библиотеку и установки ссылки на Microsoft Word 8.0 Object Library программа выполняется без ошибок.

13.1.2. Позднее связывание

Избавиться от недостатка раннего связывания можно, заранее не объявляя тип объекта, то есть, используя позднее связывание. В таком случае, для объявления объектной переменной необходимо воспользоваться наиболее общим типом, который может содержать указатель на любой объект. В VBA это типы ***Variant*** и ***Object***. Для повышения производительности рекомендуется использовать тип ***Object***.

```
Public Sub TestLinking()

    Dim WordApp As Object
    Dim Doc As Object
    Set WordApp = CreateObject("Word.Application")
    WordApp.Visible = True
```



```
    Set Doc = WordApp.Documents.Add
    Doc.Range.Text = UCase("Text in Document.")

End Sub
```

Очевидно, что при позднем связывании использовать оператор ***new*** для создания нового экземпляра объекта уже невозможно, поэтому необходимо использовать функции ***CreateObject*** или ***GetObject***.

По сравнению с ранним связыванием происходит инвертирование достоинств и недостатков. Теперь программный проект работает в любой версии Office, начиная с 97 – явной ссылки на библиотеку типов нет, а имена объектов одинаковые в каждой версии. Но при разработке программы исчезли подсказки и заметно снизилась скорость взаимодействия клиента и сервера. Также пропадает возможность обработки сообщений от объекта при помощи ключевого слова ***WithEvents***.

Принимать решение об использовании раннего или позднего ввязывания необходимо, учитывая обстоятельства. Если известно, что документ будет использоваться на конкретной версии Office, тогда желательно использовать раннее связывание. Если предполагается, что документ будет использоваться на различных версиях Office, то на этапе разработки можно использовать раннее связывание (для использования подсказок в редакторе), для создания экземпляров объектов использовать ***CreateObject***, а при передаче пользователям заменить все явно указанные типы на ***Object***.

Вопросы для самопроверки

1. В чем отличие позднего и раннего связывания? Какие возможности позднего связывания могут быть полезными при разработке приложений?
2. В чем особенность применения позднего связывания? Какие ограничения появляются при его использовании?
3. Приведите примеры использования раннего и позднего связывания при подключении приложений-серверов?

Глава 14. Общие рекомендации по разработке приложений

В данном разделе обобщаются приведенные выше рекомендации, описания доступных разработчику приложений на основе Microsoft Office средств.

При разработке приложений на базе приложений Office можно использовать следующие общие рекомендации и приемы:

- создание процедур *обработки событий* объектов, соответствующих приложению, документам, рабочим книгам и т.п., для выполнения определенных действий при возникновении соответствующего события (***New*** – создание, ***Open*** – открытие, ***Close*** и ***BeforeClose*** – закрытие и т.п.);
- реализация процедур, автоматизирующих создание документов с заданной структурой, их форматирование;
- написание обработчиков ошибок приложения для отдельных процедур или централизованная обработка ошибок;
- разработка собственных диалоговых окон для ввода и редактирования информации;
- добавление собственных меню и панелей инструментов;
- установка защиты частей документов (разделов в Word, диапазонов ячеек, рабочих листов и т.д. в Excel), содержащих информацию, которая не должна изменяться пользователем непосредственно;
- использование возможностей по скрытию частей документа, содержащих информацию, не предназначенную для просмотра пользователем (формул, строк и столбцов, рабочих листов в Excel, фрагментов текста в Word);
- написание процедур идентификации пользователя при доступе к защищенным данным;
- использование переменных, определенных пользователем, элементов управления, полей (в Word), именованных диапазонов (в Excel) для ввода информации и редактирования документов;

- проектирование транзакций, обеспечение целостности данных при работе с разделяемыми данными в сети;
- сохранение шаблонов и надстроек для совместного их использования;
- защита кода приложений от несанкционированных изменений;
- закрепление правил, соглашений по структуре документа и кодированию при совместной работе нескольких разработчиков для обеспечения возможности разделения кода.

Далее рассматриваются дополнительные рекомендации и возможности, используемые при разработке приложений.

Вопросы для самопроверки

1. Каким образом можно предотвратить «порчу» стандартного интерфейса приложений Microsoft Office в случае аварийного завершения приложений, построенных на основе пакета Microsoft Office, использующих специально разработанный собственный интерфейс?
2. Каковы возможности обработки ошибок в приложениях, разработанных на основе Microsoft Office?
3. Какими средствами можно защитить документы и приложения Microsoft Office от оштбок и несанкционированного доступа?
4. Какими средствами можно обеспечить целостность данных при работе с разделяемыми данными в сети?
5. Какие правила следует соблюдать при коллективной работе с приложениями пакета, при разработке на их основе собственных приложений?

Глава 15. Разработка справочной системы

Включение справочной системы в приложение в настоящее время является стандартом. В приложениях Windows используется два типа справочной информации: всплывающие подсказки и файлы справки.

15.1. Использование всплывающих подсказок

Всплывающие подсказки используются для вывода контекстной справки для элементов управления.

Каждый элемент управления в форме имеет свойство ***ControlTipText***. Значение этого свойства задает текст, отображаемый на экране, когда пользователь устанавливает указатель мыши на этом элементе. Этот текст и называется всплывающей подсказкой. Этот вид справки удобен, так как не требуется создавать специальный раздел в файле справки.

15.2. Контекстная справка «Что это такое?»

Контекстная справка «Что это такое?» используется для вывода краткой справочной информации. В приложениях Microsoft Office щелчок правой кнопкой мыши по элементу диалогового окна (формы) открывает контекстное меню, из которого можно выбрать пункт **Что это такое?**. Для вызова контекстной справки можно также «нажать» кнопку **Что это такое?** системного меню и щелкнуть мышью по интересующему элементу.

Для подключения контекстной справки используются два свойства: свойство ***WhatsThisButton*** формы задает вывод кнопки, которая используется для отображения контекстной справки в диалоговом окне (если значение равно ***True***, кнопка выводится в заголовке окна); свойство ***WhatsThisHelp*** определяет, используется ли для контекстной справки всплывающее окно, или же главное окно «Справка».

Если ***WhatsThisHelp*** равно ***True***, то раздел справочной системы приложения, идентификатор которого задан с помощью свойства ***HelpContextID***, выводится с помощью команды **Что это**

такое? (команда доступна через контекстное меню объекта или с помощью кнопки, отображаемой в заголовке окна).

Если ***WhatsThisHelp*** равно ***False***, то для запуска справочной системы и загрузки раздела, указанного с помощью свойства ***HelpContextID***, используется клавиша *F1*.

Метод ***WhatsThisMode*** преобразует указатель мыши к виду «Что это такое?» (как после щелчка по соответствующей кнопке в заголовке окна), т.е. готовит приложение к выводу справки для выбранного объекта в окне.

Использование контекстной справки предполагает наличие у приложения файла справки, который должен быть подготовлен разработчиком.

15.3. Подготовка файла справки

Файл справки может быть подготовлен с помощью любой утилиты, предназначенной для разработки справочников, например: RoboHelp, Doc-To-Help, Visual Help, Help Magician, ForeHelp, Microsoft Help Workshop и др.

Общая схема подготовки файла справки включает следующие шаги:

1. Подготовка текстового документа (или нескольких текстовых документов), содержащих текст справочника и сохранение его в заданном формате (обычно – RTF).
2. Разработка проекта для преобразования созданных файлов в справочник (создание содержания, редактирование ссылок и т.д.) и его компиляция в HLP-файл с помощью специальной программы.

Для облегчения этой работы в состав Instant Help включен специальный шаблон InstHelp.dot, который позволяет разрабатывать справочник в среде MS Word. В этом шаблоне содержатся определения всех необходимых стилей, используемых для разметки файла справки (новых тем, переходов и т.п.), макросы и дополнения к меню и панелям инструментов.

Файл с текстом справочника создается на основе указанного шаблона, а генератор справки, конвертирующий созданных файл в справочник, запускается с помощью команды меню или спе-

циализированной кнопки на панели инструментов. Для компиляции файла используется MS Help Workshop.

Вопросы для самопроверки

1. Каким образом задать всплывающую подсказку для элемента управления?
2. Каковы общие правила разработки контекстной спраки?

Глава 16. Дополнительные рекомендации

16.1. Установка приложения

При разработке приложений пользователь всегда должен предусматривать возможность установки приложения неквалифицированным пользователем и возможность последующих обновлений созданного приложения. Поэтому для облегчения решения этих задач необходимо разработать специальные программы установки приложений (в том числе и приложений, созданных на основе MS Office).

В общем случае программа установки должна выполнять следующие действия:

- определять папки для установки приложения и проверять возможность установки приложения в указанные папки,
- давать возможность выбора файлов для установки (если это необходимо, т.е. если, например, у пользователя есть выбор вариантов установки),
- проверять наличие нужных для установки файлов на дистрибутиве и контролировать результаты их копирования в указанные папки,
- обеспечивать возможность «отката», т.е. отказа от выполнения установки с возвратом к исходному состоянию (например, если выбранные для установки папки уже содержат файлы с такими же именами, произошла ошибка при копировании очередного файла и т.п.),

- сохранять параметры приложения в Реестре системы, создав для этого специальный подраздел в разделе Реестра «HKEY_CURRENT_USER\Software»;
- обеспечить возможность деинсталляции/обновления приложения (создать файл с информацией об установке, о размещении файлов и папок, изменениях, внесенных в стандартные папки, в реестр и т.п.) с восстановлением/обновлением информации в Реестре.

Выполнение этих требований облегчит работу с приложением, его установку и модификацию.

16.2. Повышение эффективности работы приложения

Для облегчения работы пользователя с приложением при его разработке необходимо решить следующие задачи:

- обрабатывать ошибки (инструкция ***On Error*** и специальные средства приложений, например, проверка вводимых значений в Excel), локализуя их, сообщая пользователю о том, какая ошибка произошла, и что он должен предпринять;
- отключать обновление (перерисовку) экрана при выполнении промежуточных операций (открытия файлов, обновления данных в таблицах и т.п.), что повысит скорость выполнения программы (свойства ***ScreenUpdating***, ***Visible***);
- закрывать все файлы, открытые, но уже не используемые приложением, причем принудительно сохранять при закрытии файлы, которые должны быть сохранены, и не сохранять файлы, которые не должны сохраняться (используя метод ***SaveAs*** и свойство ***Saved***);
- возвращать фокус в исходную точку (для этого необходимо запоминать текущее положение курсора, точки вставки, рамки выделения и т.п. при запуске команды);
- восстанавливать в качестве текущих диск и папку, которые были текущими до запуска приложения;
- восстанавливать параметры приложения и установки пользователя (панели инструментов и меню, масштаб,

состояние главного окна и дочерних окон, режим отображения документов, установки, доступные через команду **Параметры** меню **Сервис**).

Выполнение этих требований не только облегчит работу пользователя с приложением, но и позволит сохранить привычную для пользователя среду.

Вопросы для самопроверки

1. Перечислите основные функции, которые должна выполнять программа установки приложения.
2. С помощью каких средств можно обрабатывать ошибки и предотвращать их? Приведите примеры.
3. В каких ситуациях следует отменять перерисовку экрана и скрывать окна активизируемых объектов? Каким образом это можно сделать? Приведите примеры.
4. С помощью каких средств можно управлять файлами, используемыми приложениями (сохранять в них их обновления, предотвращая потерю, и отменять сохранения, если должны быть сохранены неизмененные данные)? Приведите примеры.
5. Каким образом можно организовать возврат в папку, открытую при запуске приложения, после его завершения (штатного или аварийного)? Приведите соответствующий код на VBA.
6. Каким образом можно переместить фокус? Что нужно сделать для возврата фокуса в исходное положение после завершения выполнения операции? Приведите примеры кода на VBA.
7. Какими средствами можно восстановить исходные настройки приложения (параметры, настройки интерфейса (меню, панелей, окон) и пр.) после выполнения офисной программы, разработанной на его основе?

Приложение 1. Автоматизированное рабочее место оператора автозаправочной станции

Постановка задачи

Рассмотрим пример разработки на базе Excel приложения, автоматизирующего работу оператора автозаправочной станции (АЗС).

На АЗС производится продажа бензина (АИ-76, АИ-92, АИ-95) и дизельного топлива (ДТ) с помощью 8 колонок. Для каждого из вида топлива имеется резервуар вместимостью 20 000 литров. В обязанности оператора входят продажа топлива, заказ топлива соответствующего вида при его окончании. Таким образом, в программе необходима реализация следующих функций:

- организация продажи топлива: возможность изменения отпускной цены, организация расчета с клиентом, организация сменной работы;
- контроль наполнения резервуаров: изменение информации при поступлении топлива, выдача предупредительных сообщений при критических объемах топлива;
- ведение информации о персонале станции.

Разработка структуры рабочей книги

Рабочая книга приложения будет состоять из трех листов:

- главного листа, с помощью элементов которого будет реализована основная функциональность приложения;
- служебного листа, содержащего справочную информацию;
- листа, содержащего информацию о работниках станции.

На главном листе будет размещена основная информация для пользователя, элементы управления и команды, обеспечивающие основные функции приложения.

Второй лист будет содержать справочную и служебную информацию, он будет скрыт от пользователя.

На третьем листе поместим информацию о персонале, он также будет скрыт от пользователя, доступ к нему будет организован только через экранную форму.

Подготовим главный лист приложения, поместив на него таблицу, которая будет содержать текущие цены на топливо и диаграмму, которая будет отображатьть состояние резервуаров, а также командные кнопки для «продажи» топлива.

Добавим на рабочий лист таблицу цен на топливо, как показано ниже.

Тип топлива	*Цена за литр*
Аи-76	10,00р.
Аи-92	14,00р.
Аи-95	16,00р.
ДТ	12,00р.

Диапазону цен из четырех ячеек, выделенных цветом, присвоим имя *rgFuelPrice*, что упростит обращение к этим данным.

На второй лист книги поместим таблицу «Резервуары», содержащую информацию о состоянии резервуаров:

Тип топлива	*Вместимость (л)*	*Заполнено (л)*
Аи-76	20000	14982
Аи-92	20000	16139
Аи-95	20000	14900
ДТ	20000	6000

На базе таблицы «Резервуары», размещенной на втором листе, построим диаграмму, вид которой показан на рис. А1.1. Разместим эту диаграмму на первом листе:

На второй лист поместим также небольшой справочник, данные которого будут использованы при работе экранных форм:

Должности
Кассир
Оператор
Техник

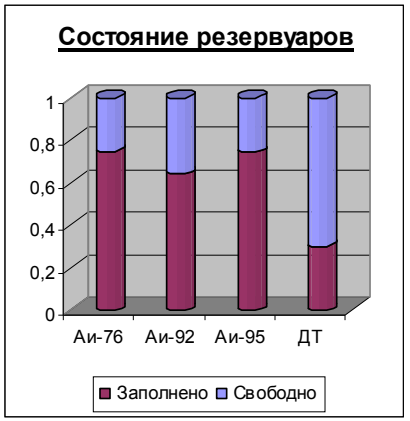


Рис. А1.1. **Диаграмма состояния резервуаров**

Разместим на главном рабочем листе 8 командных кнопок для «продажи» топлива на соответствующей колонке. Для этого:

– с помощью команд меню **Вид** необходимо отобразить панель «Элементы управления»;
– нажать на ней кнопку «Режим конструктора»;
– разместить элементы управления – командные кнопки (для размещения кнопки на панели «Элементы управления» выбирается соответствующий элемент (прямоугольник) и кнопка «рисуется» на рабочем листе, как обычная автофигура с помощью элементов панели рисования).

Первый лист рабочей книги примет вид показанный на рис. А1.2.

На третьем листе книги поместим информацию о персонале автозаправочной станции, как это показано в следующей таблице:

Фамилия	*Имя*	*Отчество*	*Пол*	*Дата рождения*	*Должность*	*Разряд*
Иванов	Сидор	Петрович	Мужской	03.12.1980	Кассир	1
Петров	Иван	Сидорович	Мужской	04.12.1980	Кассир	2

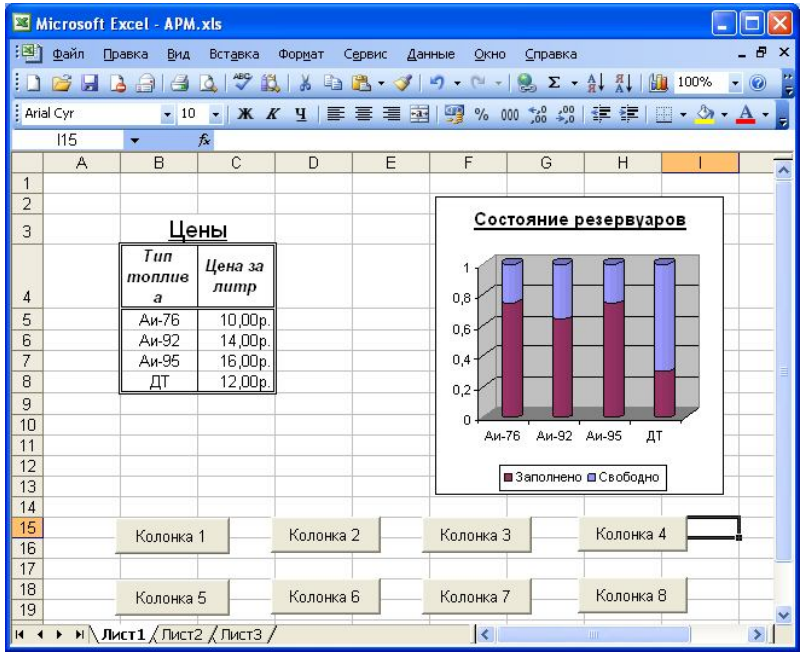


Рис. А1.2. **Размещение объектов на главном листе приложения**

Мы разработали структуру рабочей книги, подготовив все необходимое для следующих этапов разработки нашего приложения.

Настройка пользовательского интерфейса

Прежде всего, необходимо заменить стандартный интерфейс Excel, добавив команды меню и панели инструментов, необходимые для данного приложения, и удалив неиспользуемые команды, строку формул, заголовки строк и столбцов, ярлычки листов.

Скроем видимые панели инструментов, обратившись к свойству ***Enabled*** панелей инструментов. При установке данного свойства в значение ***False*** панель скрывается с экрана и удаляется из списка панелей, которые пользователь может отобразить при выполнении команды «Настройка…». В приведенном ниже

коде скрываются все панели приложения. Данный способ эффективнее, чем обращение к свойству ***Visible*** панели инструментов, т.к. при завершении работы приложения при установке ***Enabled*** в ***True*** будут восстановлены именно те панели, которые были видны при запуске программы. Для выполнения указанных действий программным способом необходимо написать следующий код на VBA, добавив его в текст разрабатываемого макроса, который будет запускаться при открытии рабочей книги «APM.xls» (см. ниже), на основе которой создается приложение:

```
Dim cb As CommandBar
For Each cb In Application.CommandBars
    cb.Enabled = False
Next cb
```

Далее удалим стандартные элементы интерфейса Excel. Основные объекты, к которым мы обращаемся, – это объект ***Application***, представляющий приложение в целом, и объект ***ActiveWindow***, который соответствует окну рабочей книги. Данный код лучше оформить в виде отдельной процедуры, получающей в качестве параметра логическое значение – отобразить или убрать соответствующие элементы управления:

```
With Application
    .DisplayFormulaBar = False
    .DisplayScrollBars = False
    .DisplayStatusBar = False
    .Caption = "APM"
    With .ActiveWindow
        .DisplayGridlines = False
        .DisplayWorkbookTabs = False
        .DisplayHeadings = False
        .DisplayHorizontalScrollBar = False
        .DisplayVerticalScrollBar = False
        .Caption = "Управление АЗС"
    End With
End With
```

Добавим панель инструментов и разместим необходимые команды. Ниже представлен внешний вид панели инструментов (рис. А1.3) и программный код, создающий данную панель:

```
Set cmdbar = Application.CommandBars.Add _
                                ("Management", msoBarTop)
```

```
Application.CommandBars.DisableCustomize = True
Dim but As CommandBarButton
With cmdbar
    .Protection = msoBarNoCustomize
    Set but = .Controls.Add(msoControlButton)
    but.OnAction = "Getgaz"
    but.Style = msoButtonCaption
    but.Caption = "Получить бензин"
    Set but = .Controls.Add(msoControlButton)
    but.OnAction = "Sellgaz"
    but.Style = msoButtonCaption
    but.Caption = "Продать бензин"
    Set but = .Controls.Add(msoControlButton)
    but.OnAction = "Changeprice"
    but.Style = msoButtonCaption
    but.Caption = "Сменить цены"
…
End With
```

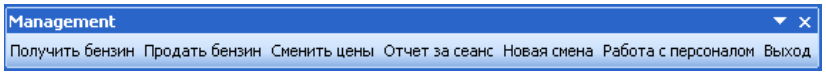


Рис. А1.3. **Панель инструментов приложения**

Первоначально к коллекции *CommandBars* мы добавляем новый элемент – новую панель инструментов. Коллекция Controls добавленной панели инструментов содержит все элементы управления, расположенные на данной панели. Таким образом, для добавления новых кнопок необходимо добавить к данной коллекции новый элемент. Кроме обычных кнопок, на панели инструментов могут находиться более сложные элементы управления (см. таблицу).

Значение Type	***Возвращаемый класс объекта***
msoControlButton (командная кнопка)	CommandBarButton
msoControlEdit (поле редактирования)	CommandBarComboBox
msoControlDropdown (выпадающий список)	CommandBarComboBox
msoControlComboBox (комбинированный список)	CommandBarComboBox
msoControlPopup (контекстное меню)	CommandBarPopup

Действия, выполняемые командой, должны быть запрограммированы в виде процедуры. Для передачи информации в исполняемый макрос можно использовать параметр **Parameter**, задаваемый при создании элемента или, что удобнее, свойство ***Parameter***. Свойство ***OnAction***, которым обладают все элементы панели, позволяет связать элемент с исполняемым макросом. Формально свойство является строкой, задающей имя макроса, который и будет вызываться в ответ на выбор пользователя.

Описанные выше блоки кода необходимо поместить в обработчик события ***Workbook_Open***, вызываемый при открытии рабочей книги. Стандартный интерфейс Excel будет заменяться каждый раз при открытии рабочей книги на интерфейс, созданный разработчиком. Очевидно, что при закрытии книги (событие ***Workbook_BeforeClose***) необходимо выполнить обратные действия – восстановить исходный интерфейс. Приложение должно руководствоваться принципом «Меня здесь не было», т.е. полностью восстановить все параметры, которые оно изменило. Иначе у пользователя, работающего с нашим приложением и незнающего VBA, могут возникнуть значительные проблемы, например, при исчезновении главного меню офисной программы.

Разработка экранных форм

Следующим шагом в построении нашего приложения будет разработка экранных форм и программного кода, поддерживающего их функционирование. Использование экранных форм дает значительные преимущества. Работая с формой, пользователь не может нарушить целостность данных, изменить структуру таблиц. Также при использовании форм возможна интерактивная проверка данных, введенных пользователем, можно организовать проверку условий любой сложности. Применение специальных элементов управления позволяет упростить ввод и редактирование данных.

Добавим форму «Отпускные цены», служащую для изменения текущих отпускных цен на топлива. Разместите элементы управления как показано на рис. А1.4. При нажатии на кнопку «Изменить» прежде всего надо организовать проверку введенных пользователем данных, затем сохранить их в диапазоне **rgFuelPrice**.

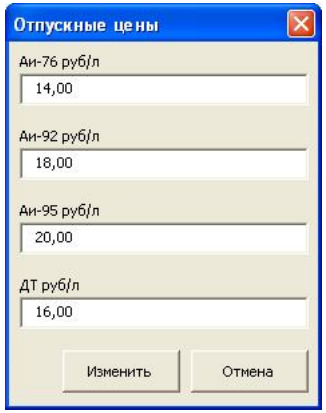


Рис. А1.4. **Экранная форма «Отпускные цены»**

```
Private Sub cmdChange_Click()
    If Not IsNumeric(txtPrice76.Text) Then
        SetFocusToError txtPrice76
        Exit Sub
    End If
    If Not IsNumeric(txtPrice92.Text) Then
        SetFocusToError txtPrice92
        Exit Sub
    End If
    If Not IsNumeric(txtPrice95.Text) Then
        SetFocusToError txtPrice95
        Exit Sub
    End If
    If Not IsNumeric(txtPriceDT.Text) Then
        SetFocusToError txtPriceDT
        Exit Sub
    End If
    With Worksheets("Главный").Range("rgFuelPrice")
        .Cells(1, 1).Value = CDbl(txtPrice76.Text)
        .Cells(2, 1).Value = CDbl(txtPrice92.Text)
        .Cells(3, 1).Value = CDbl(txtPrice95.Text)
        .Cells(4, 1).Value = CDbl(txtPriceDT.Text)
    End With
End Sub
Public Sub SetFocusToError(ctrl As Control)
    ctrl.SelStart = 0
    ctrl.SelLength = Len(ctrl.Text)
    ctrl.SetFocus
End Sub
```


С помощью встроенной функции ***IsNumeric*** мы производим проверку, можно ли рассматривать введенное значение как число. Если нет, то производится вызов специально разработанной функции **SetFocusToError**, которая устанавливает фокус на элемент управления с неверным значением. Также в код необходимо добавить вывод на экран сообщения об ошибке с помощью функции ***MsgBox***. При проверке всех значений данные сохраняются на рабочем листе в требуемом диапазоне.

Добавим еще один обработчик события для данной формы: при открытии формы необходимо загрузить текущие значения цен в элементы управления:

```
Private Sub UserForm_Initialize()
    With Worksheets("Главный").Range("rgFuelPrice")
        txtPrice76.Text = .Cells(1, 1).Text
        txtPrice92.Text = .Cells(2, 1).Text
        txtPrice95.Text = .Cells(3, 1).Text
        txtPriceDT.Text = .Cells(4, 1).Text
    End With
End Sub
```

Подобным образом строятся остальные формы изменения данных на рабочих листах, основные действия, которые необходимо реализовать на форме: загрузка текущих данных, контроль данных и сохранение новых данных.

Экранная форма «Продажа» (рис. А1.5) вызывается при нажатии на одну из восьми кнопок. За каждой их колонок закреплен определенный вид топлива.

При продаже топлива должна измениться информация в таблице «Резервуары», находящейся на втором листе. С данной таблицей связана диаграмма, таким образом, оператор сможет визуально контролировать объем топлива, находящегося в резервуарах.

Форма позволяет либо ввести необходимое клиенту количество топлива в литрах (в этом случае автоматически вычисляется цена), либо указать сумму, на которую он совершает покупку (в этом случае количество литров также вычисляется автоматически). Перед продажей топлива необходимо проверить условие, есть ли в резервуаре необходимое количество топлива.

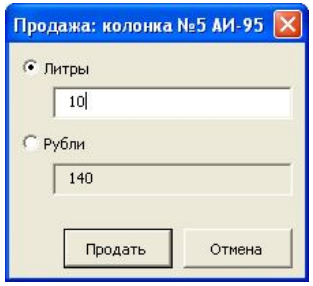


Рис.А1.5. **Экранная форма «Продажа»**

На форме «Получение бензина», показанной ниже (рис. А1.6), располагаются выпадающий список и текстовое поле. При доставке бензина соответствующего типа обновляется информация в таблице «Резервуары». При получении топлива необходимо проверить, достаточно ли в резервуаре свободного места.

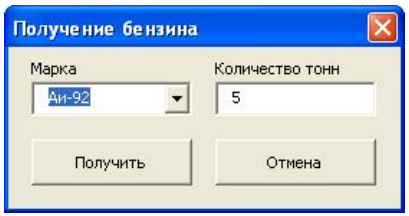


Рис.А1.6. **Экранная форма «Получение бензина»**

Разработаем экранные формы для работы с информацией о сотрудниках.

Форма «Персонал» (рис. А1.7) содержит элемент управления ***ListBox*** и четыре командные кнопки. Для того чтобы список отображал несколько колонок, необходимо изменить свойство ***ColumnCount***: присвоим данному свойству значение 7. Для настройки ширины столбцов используется свойство ***ColumnWidths***, в нашем случае установим значение «49.95 pt; 35 pt; 49.95 pt; 45 pt; 67.95 pt; 60 pt; 40 pt». Для отображения заголовков столбцов необходимо установить свойство ***ColumnHeads*** в ***True***.

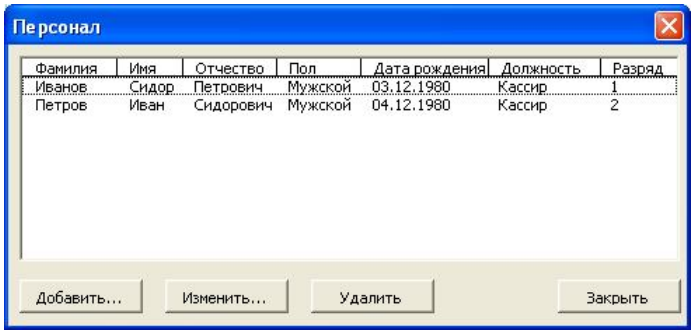


Рис.А1.7. **Экранная форма «Персонал»**

Команды «Добавить», «Изменить» и «Удалить» используются для изменения данных на листе «Персонал». Таким образом, нам необходимо программно добавлять, редактировать и удалять строки таблицы. Программный код формы представлен ниже.

```
Public Sub UpdateListRowSource()
    Dim ws As Worksheet
    Set ws = Worksheets("Персонал")
    lstEmloyees.RowSource = "'Персонал'!A2:G" + _
                  CStr(ws.UsedRange.SpecialCells _
                  (xlCellTypeLastCell).Row)
End Sub

Private Sub cmdAdd_Click()
    frmEmployee.Show
    If frmEmployee.Tag = "OK" Then
        Dim NewRow As Integer
        Dim r As Range, ws As Worksheet
        Set ws = Worksheets("Персонал")
        Set r = ws.UsedRange.SpecialCells _
                              (xlCellTypeLastCell)
        NewRow = r.Row + 1
        SaveDataToRow NewRow
        UpdateListRowSource
    End If
End Sub

Private Sub SaveDataToRow(RowIndex As Integer)
    With frmEmployee
        Dim ws As Worksheet
```

```
        Set ws = Worksheets("Персонал")
        ws.Cells(RowIndex, 1).Value = _
                              .txtLastname.Text
        ws.Cells(RowIndex, 2).Value = .txtName.Text
        ws.Cells(RowIndex, 3).Value = _
                              .txtPatronymic.Text
        ws.Cells(RowIndex, 4).Value = .cmbSex.Text
        ws.Cells(RowIndex, 5).Value = _
                              .dtpDateOfBirth.Value
        ws.Cells(RowIndex, 6).Value = _
                              .cmbAppointment.Text
        ws.Cells(RowIndex, 7).Value = .txtGrade.Text
    End With
End Sub

Private Sub cmdClose_Click()
    Me.Hide
End Sub

Private Sub cmdEdit_Click()
    Dim RowIndex As Integer, ws As Worksheet
    RowIndex = lstEmloyees.ListIndex + 2
    Set ws = Worksheets("Персонал")
    With frmEmployee
        .txtLastname.Text = _
                        ws.Cells(RowIndex, 1).Value
        .txtName.Text = ws.Cells(RowIndex, 2).Value
        .txtPatronymic.Text = _
                        ws.Cells(RowIndex, 3).Value
        .cmbSex.Text = ws.Cells(RowIndex, 4).Value
        .dtpDateOfBirth.Value = _
                        ws.Cells(RowIndex, 5).Value
        .cmbAppointment.Text = _
                        ws.Cells(RowIndex, 6).Value
        .txtGrade.Text = ws.Cells(RowIndex, 7).Value
        .Show vbModal
    End With
    SaveDataToRow RowIndex
End Sub

Private Sub cmdDelete_Click()
    If lstEmloyees.ListIndex <> -1 Then
        Dim RowIndex As Integer, ws As Worksheet
        RowIndex = lstEmloyees.ListIndex + 2
        Set ws = Worksheets("Персонал")
        ws.Rows(RowIndex).Delete
        UpdateListRowSource
```



```
    End If
End Sub

Private Sub UserForm_Initialize()
    UpdateListRowSource
End Sub
```

Процедура **UpdateListRowSource** используется для обновления свойства **RowSource** списка. При добавлении и удалении строк адрес связанного диапазона необходимо обновить. Строка

```
ws.UsedRange.SpecialCells(xlCellTypeLastCell).Row
```

используется для быстрого поиска первой пустой строки. Для поиска первой пустой строки можно применить и другие методы: например, просмотреть все строки, начиная с первой, пока не будет найдена первая пустая строка. Однако, при больших объемах данных на поиск строки будет тратиться продолжительное время.

При добавлении новой строки отображается форма для ввода данных, затем с помощью рассмотренного выше метода находится первая пустая строка и с помощью специально разработанной процедуры **SaveDataToRow** данные с формы переносятся в таблицу. После добавления строки обновляется свойство **RowSource** списка и в нем появляются данные о новом работнике.

Индекс редактируемой строки в таблице определяется с помощью индекса выделенного в данный момент элемента списка, после чего данные загружаются в форму. После изменения данных обновление свойства **RowSource** списка не требуется, т.к. адрес диапазона не изменился. Список на форме будет обновлен автоматически.

При удалении строки с помощью метода ***Delete*** из коллекции ***Rows*** удаляется строка с указанным индексом, после чего необходимо изменить свойство ***RowSource*** списка.

Добавьте новую форму «Работник» (рис. А1.8). Разместите на ней три текстовых поля для фамилии, имени и отчества работника. Для выбора пола и должности работника добавьте два списка. Первый список мы программно заполним данными в обработчике события инициализации формы. Свойству **RowSource** второго списка присвойте адрес диапазона, содержащий список должностей (например, **'Данные'!B3:B6**).

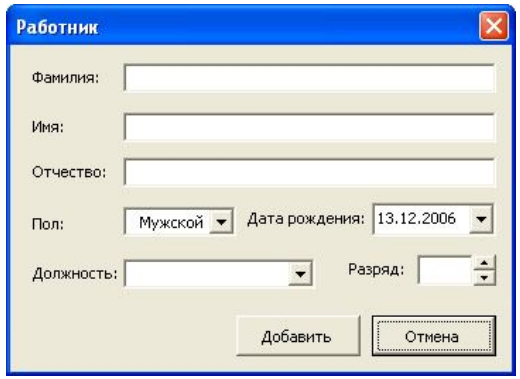


Рис. А1.8. **Экранная форма для создания/изменения данных сотрудника**

Для ввода даты будем использовать специальный элемент управления «Microsoft Date and Time Picker Control». Данный элемент управления предлагает пользователю удобный способ работы с датами. По умолчанию данного элемента нет на панели элементов управления (***Toolbox***), поэтому перед использованием его необходимо туда вынести. Для этого в контекстном меню панели элементов управления выберите пункт «Additional Controls…», затем в списке «Available Controls» выберите пункт «Date and Time Picker Control 6.0». На панели элементов управления появится новая пиктограмма, соответствующая добавленному элементу управления.

Для выбора разряда работника будем использовать элемент управления **SpinButton**, позволяющий выбирать числовые значения из заданного диапазона с указанным шагом. Разместите на форме текстовое поле, а рядом с ним элемент управления **SpinButton**, установите свойство **Min** равным 1, а свойству **Max** присвойте значение 5. Программный код формы представлен ниже.

```
Option Explicit
Public Mode As Integer
Private Sub cmdCancel_Click()
    Me.Tag = "Cancel"
    Me.Hide
End Sub
```



```
Private Sub spnGrade_Change()
    txtGrade.Value = spnGrade.Value
End Sub

Private Sub UserForm_Initialize()
    cmbSex.AddItem "Мужской"
    cmbSex.AddItem "Женский"
    cmbSex.ListIndex = 0
End Sub

Private Sub cmdSave_Click()
    Me.Tag = "OK"
    If Trim(txtLastname.Text) = "" Then
        MsgBox "Ошибка при заполнении поля: Фамилия."
        Me.Tag = "Cancel"
    End If

    If Trim(txtName.Text) = "" Then
        MsgBox "Ошибка при заполнении поля: Имя."
        Me.Tag = "Cancel"
    End If

    If Trim(txtPatronymic.Text) = "" Then
        MsgBox "Ошибка при заполнении поля: Отчество."
        Me.Tag = "Cancel"
    End If
    Me.Hide
End Sub

Public Sub SetFocusToError(ctrl As Control)
    ctrl.SelStart = 0
    ctrl.SelLength = Len(ctrl.Text)
    ctrl.SetFocus
End Sub
```

Приложение 2. Автоматизация заполнения бланка с помощью программы текстового процессора Word

Постановка задачи

Разработать приложение, автоматизирующее заполнение бланка удостоверения о краткосрочном повышении квалификации. Программа должна предоставить пользователю экранную форму для заполнения полей бланка и вывести эти данные в соответствующие поля бумажного бланка при печати. Необходимо также вести учет выданных удостоверений в специальном журнале.

Разработка документа

Одна из основных трудностей в данном приложении – это вывод данных в поле бланка. Пользователь вставляет в принтер чистый бланк удостоверения, а программа должна вывести необходимые данные в точно установленные позиции. Ошибка при расположении информации может испортить бланк, количество которых ограничено. Также надо учитывать возможность переноса длинных строк и другие особенности расположения текста. Можно предложить множество вариантов решения данной проблемы, мы остановимся на одном из самых простых и эффективных, позволяющем за минимальное время получить наилучший результат.

Основная идея подхода заключается в том, что мы будем использовать отсканированное изображение бланка, на которое наложим таблицу. Параметры абзаца каждой из ячеек таблицы будут настроены таким образом, что при вводе в них текста, он займет правильно положение на бланке. После настройки таблицы фоновое изображение можно удалить и не отображать границы таблицы. При печати документа на реальном бланке текст попадет на требуемое место, так как его расположение было точно настроено по «образу» бланка.

Начнем выполнение задания. Прежде всего установим размер бумаги документа, соответствующий размеру бланка. Для этого необходимо выбрать команду **Файл ▸ Параметры страницы…** и на вкладке «Размер бумаги» в раскрывающемся списке установить значение «другой», а в полях «Высота» и «Ширина» установить соответствующие числовые значения. В рассматриваемом примере, бланк имеет стандартные размеры – формат А4, требуется лишь установить альбомную ориентацию страницы на вкладке «Поля» и ширину полей.

Затем нам необходимо «вставить» в документ отсканированный бланк. Очень важно, чтобы копия была сделана в масштабе 1:1, то есть бланк и его копия имели одинаковые размеры, в противном случае нам не удастся правильно разместить текст при печати. Для вставки рисунка необходимо выполнить команду **Формат ▸ Фон ▸ Подложка…**, выбрать вариант «Рисунок», указать путь к файлу рисунка и назначить 100% масштаб (рис. А2.1). Положение рисунка можно изменить, используя команду «Колонтитулы», в режиме работы с колонтитулами фоновый рисунок доступен для изменения.

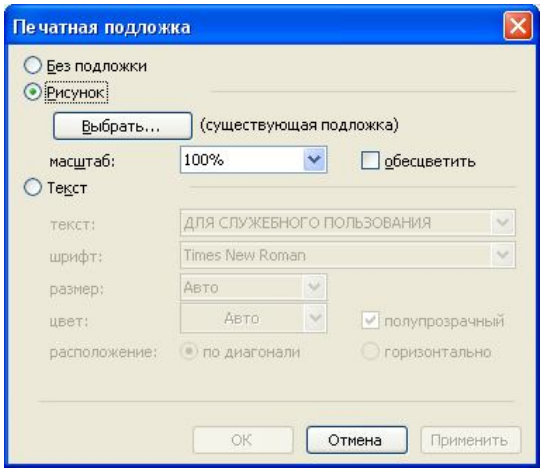


Рис. А2.1. **Диалоговое окно настройки печатной подложки**

Вставьте в документ таблицу, в которой будет располагаться текст, «впечатываемый» в бланк (рис. А2.2). Если текст бланка состоит из нескольких несвязных блоков, то необходимо вставить в документ несколько таблиц.

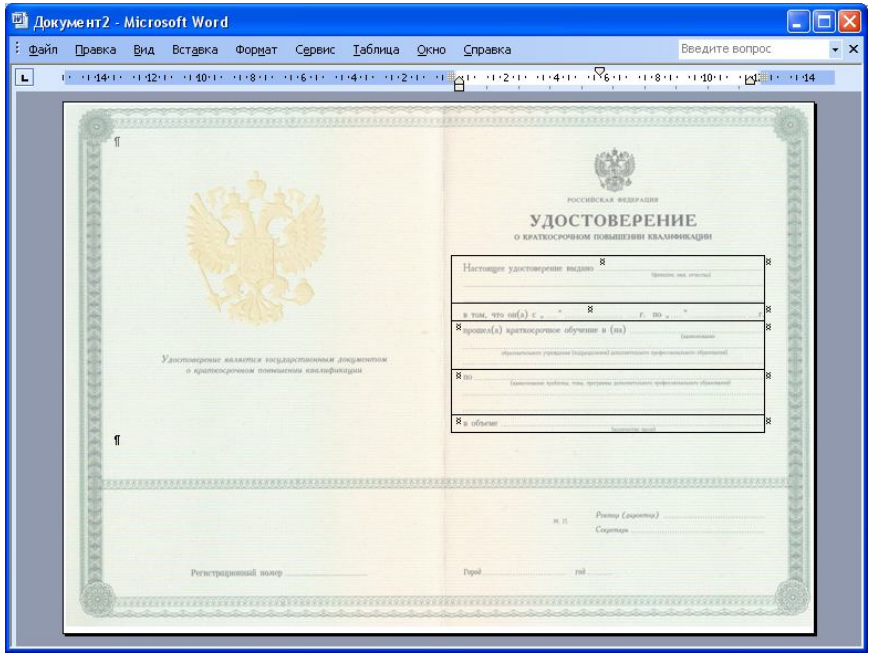


Рис.А2.2. **Внешний вид документа после помещения фона и таблицы**

После размещения таблиц необходимо приступить к их настройке. Абзац каждой из ячеек таблицы необходимо настроить таким образом, чтобы вводимый в ячейку текст попал точно в стоку бланка, причем необходимо учесть возможность переноса строк. Для этого необходимо настроить абзацные отступы и межстрочный интервал (рис. А2.3).

Следующим этапом разработки документа будет добавление *раздела со справочной информацией*, в котором будет храниться список, содержащий название проводимых курсов. Список оформим в виде таблицы, вставленной в текст (рис. А2.4).

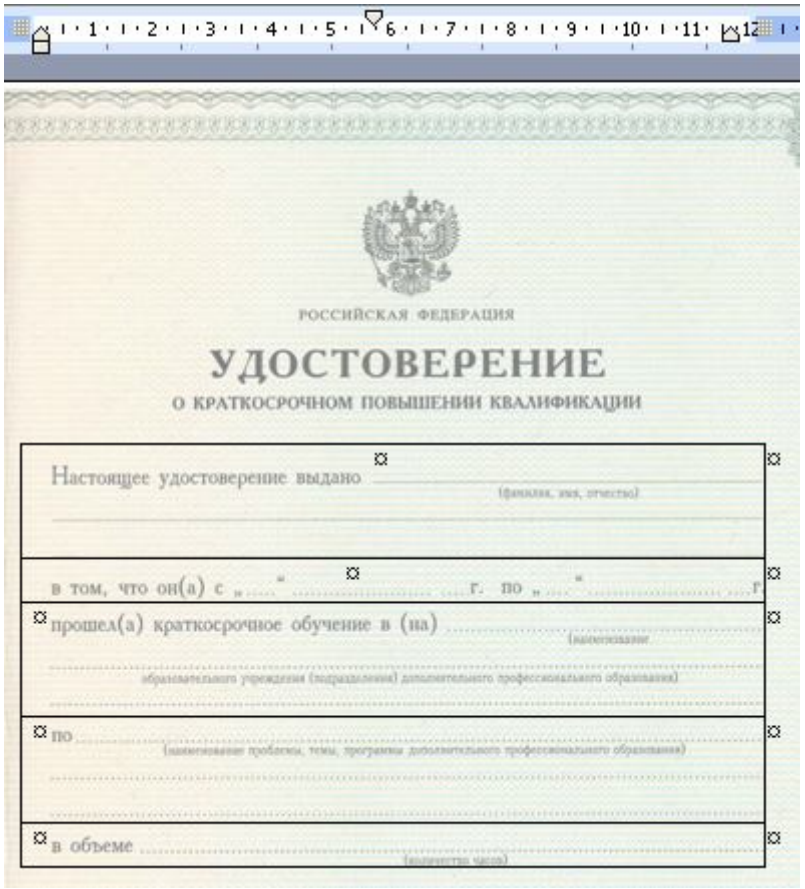


Рис. А2.3. **Размещение и настройка таблицы**

Название курсов
Английский для начинающих
Параллельное программирование
Офисные технологии

Рис. А2.4. **Таблица, используемая для ввода данных в бланк**

Для облегчения программного доступа к таблице наложим на нее закладку. Для этого выделим таблицу полностью и выполним команду **Вставка ▸ Закладка…**. В поле «Имя закладки» (рис. А2.5) введем имя «СписокКурсов» (без пробела – при вводе имен используются ограничения).

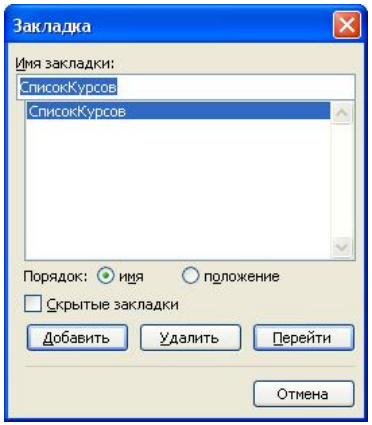


***Рис.А2.5.* Диалоговое окно вставки закладки**

Справочная информация, сохраняемая в отдельном разделе, после разработки приложения может быть скрыта от пользователя. Ознакомьтесь с возможностями «скрытия» по справочной системе приложения. В данном случае справочную информацию можно оставить в тексте документа, учитывая это при печати – на печать следует выводить только «бланк».

Разработка экранной формы

Экранная форма приложения должна предоставить возможность ввода данных в поля документа и проверку корректности этих данных (рис. А2.6). Для разработки формы необходимо переключится в среду программирования VBA с помощью комбинации клавиш *Alt+F11*.

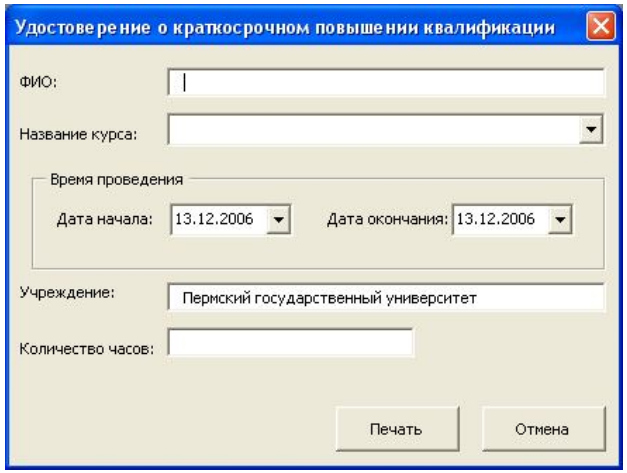


***Рис. А2.6.* Внешний вид экранной формы**

Программный код формы представлен ниже.

```
Private Sub cmdОтмена_Click()
    Me.Hide
End Sub

Private Sub cmdПечать_Click()
  With ThisDocument.Tables(1)
   .Cell(1, 1).Range.Text = txtФИО.Text
   .Cell(2, 1).Range.Text = CStr(dtpДатаНачала.Value) _
                          + CStr(dtpДатаОкончания.Value)
   .Cell(3, 1).Range.Text = txtУчреждение.Text
   .Cell(4, 1).Range.Text = cmbНазваниеКурса.Text
   .Cell(5, 1).Range.Text = txtКоличествоЧасов.Text
  End With
  Application.PrintOut FileName:="", _
              Range:=wdPrintRangeOfPages, _
              Item:=wdPrintDocumentContent, _
              Copies:=1, Pages:="1", _
              PageType:=wdPrintAllPages, _
              ManualDuplexPrint:=False, Collate:=True, _
              Background:=True, PrintToFile:=False, _
              PrintZoomColumn:=0, PrintZoomRow:=0, _
              PrintZoomPaperWidth:=0, _
              PrintZoomPaperHeight:=0
End Sub
```

```
Private Sub UserForm_Initialize()
    Dim i As Integer
    With ThisDocument.Bookmarks("СписокКурсов"). _
                                Range.Tables(1)
        For i = 2 To .Rows.Count
            cmbНазваниеКурса.AddItem _
                           .Cell(i, 1).Range.Text
        Next
    End With
End Sub
```

В процедуре **UserForm_Initialize** производится загрузка данный из таблицы «Список курсов». При нажатии на кнопку печати данные с формы переносятся в соответствующие ячейки таблицы. Команда **Application.PrintOut** выводит первую страницу документа на печать. В эту процедуру следует также добавить проверку введенных пользователем данных.

Приложение 3. Система автоматизации подготовки документов на основе пакета Microsoft Office

Ниже представлено законченное приложение, позволяющее вести подготовку первичных и сводных отчетов, разрабатывать иерархически организованные системы взаимосвязанных документов. Описаны общие принципы его создания и использованные приемы прогаммирования на основе приложений MS Office.

Данное приложение может служить примером разработки системы автоматизированных документов на основе офисного пакета.

Введение

Для всех крупных иерархически организованных структур (министерств, ведомств, крупных фирм) актуальна проблема получения информации и отчетности от подчиненных или подведомственных им предприятий.

В настоящий момент эта задача очень часто решается в основном вручную, путем заполнения соответствующих форм отчетности с последующей передачей их на вышестоящий уровень для обработки. При использовании традиционных методов сбора отчетности технология работы может быть описана следующим образом. Сначала линейные подразделения формируют формы отчетности с использованием каких-либо офисных приложений, например Microsoft Excel или Microsoft Word, распечатывают их и отправляют вышестоящему подразделению по факсу или с курьером. Соответствующие отделы головного подразделения получают эти формы, повторно вводят их в компьютер, рассчитывают сводные показатели, и только после этого информация может быть доступна руководству для принятия управляющих решений. Ситуация усугубляется, если структура многоуровневая. В такой структуре описанный выше цикл может повторяться несколько раз.

Очевидно, все вышеперечисленные действия требуют больших затрат и рутинной работы. Сроки прохождения информации по такой схеме составляют значительное время (в зависимости от глубины уровней иерархии и сложности отчетов, а также территориальной распределенности подотчетных предприятий). Если при заполнении форм возникают ошибки (особенно при заполнении полностью в ручном режиме, без всяких механизмов контроля), цикл может повторяться, либо информация поступает в искаженном виде.

Эффективное решение данной проблемы возможно с применением корпоративных многоуровневых систем. Однако реально внедрение корпоративных систем, которые позволяли бы централизовать подобную информацию, – крайне сложное дело в многоуровневой системе и может растянуться на годы. Для создания и функционирования таких корпоративных многоуровневых систем используются мощные и дорогостоящие инструментальные средства. Инструменты, разработанные для крупных предприятий, не подходят для небольших предприятий ввиду сложности их эксплуатации и высокой стоимости. На сегодняшний день на рынке программного обеспечения практически нет инструментальных средств, автоматизирующих процесс документооборота на небольшом предприятии.

В сложившейся ситуации необходимо иметь достаточно простую систему, которая позволила бы быстро и эффективно решать конкретные задачи. Эта система должна быть дешевой в эксплуатации и проста в развертывании.

Без использования автоматизированных систем после подготовки первичных документов, для извлечения из них нужной информации, ее обработки и анализа приходится ставить задачу перед программистами либо производить все манипуляции вручную. Если все же работа с документами автоматизируется, то обычно такие системы создаются программистами «с нуля», программный код ориентирован на данные конкретного документа и зачастую повторное его использование невозможно. Практика показывает, что большая часть из разрабатываемых программ автоматизации документов имеет одинаковую функциональность с некоторой поправкой на специфику предметной области. Данный факт приводит к идее создания универсального набора программных средств, автоматизирующего базовые операции с документами без вмешательства программиста.

Очевидно, что информационные системы небольших предприятий могут быть построены на базе пакета офисных программ. Современные офисные пакеты имеют развитые инструменты разработки, которые позволяют реализовать программные проекты практически любой степени сложности, использовать возможности самого пакета и функции операционной системы. Если за основу взять популярный офисный пакет, то от пользователей на стороне клиента не потребуется даже установка специального программного обеспечения. Например, за основу создания системы можно взять пакет офисных программ Microsoft Office, так как его в своей повседневной работе над различными документами применяет большое количество пользователей, в том числе и корпоративных. Данный пакет предлагает широкие возможности автоматизации работы над документом и является практически идеальной платформой для создания системы документооборота небольшого предприятия.

Одним из возможных путей решения описанной выше проблемы может быть разработка набора инструментов – CASE-средства, позволяющего автоматизировать создание систем документов, при применении которого вмешательство программиста в программный проект документа не потребуется.

Представленный здесь пакет инструментальных средств оперативной разработки офисных приложений получил название Office CASE. Office CASE – это средство, позволяющее автоматизировать подготовку отчетов и их обработку в многоуровневых иерархически организованных информационных системах

на основе использования средств стандартного выпуска пакета офисных программ Microsoft Office. Основная задача пакета – автоматизация подготовки данных и формирования на их основе первичных и сводных отчетов.

Требования к системе Office CASE

Office CASE – это набор инструментальных средств, позволяющих автоматизировать подготовку отчетов и их обработку в многоуровневых иерархически организованных информационных системах на основе использования средств стандартного выпуска пакета офисных программ Microsoft Office.

Основная задача пакета – автоматизация подготовки данных и формирования на их основе первичных и сводных отчетов, поэтому при разработке существенными являлись требования:

1) интеллектуальность документа;
2) однократный ввод данных и их совместное использование;
3) дружественный интерфейс – «ориентация на человека»;
4) мобильность;
5) независимость от версии Microsoft Office;
6) поддержка языка разметки документов XML.

Рассмотрим каждое из перечисленных выше требований более подробно.

Интеллектуальность документа

Разработанный с помощью Office CASE документ – не просто пассивный контейнер для данных, а документ, обладающий способностью выполнять различные операции над содержащимися в нем данными. Например, документ должен предоставить пользователю экранную форму для ввода и редактирования данных. Если предоставить возможность редактирования непосредственно документа, то неопытный пользователь может нарушить

структуру документа, и в этом случае будет практически невозможно организовать контроль корректности введенных данных и их обработку.

Каждый элемент системы (документ) содержит информацию о своем положении в иерархии документов, с помощью своего программного проекта реализует бизнес-логику, связанную с конкретным документом. Например, документ, являющийся родительским по отношению к другим элементам документооборота, обладает информацией о дочерних объектах. При создании дочерних документов он включает в них информацию, по которой сможет судить, что именно он является родительским документом, где располагается информация, которую следует включить в сводный отчет.

Функционирование системы документов основано на использовании метаданных, включаемых в документ. Именно включение метаданных в документ позволяет объединить в одном документе данные и методы их обработки.

Однократный ввод данных и их совместное использование

Принцип однократного ввода данных и их совместного использования является стандартом для современных информационных систем. Использование данного подхода существенно снижает количество ошибок при вводе данных и позволяет использовать полученную информацию повторно. Если мы позиционируем Office CASE как систему управления документами в распределенных средах, то система должна полностью поддерживать данный принцип.

Для реализации этого требования система должна предоставить гибкую схему работы с классификаторами. Система должна поддерживать возможность импорта существующих и создания новых классификаторов. Причем необходимо обеспечить им-

порт из различных источников данных, начиная от электронных таблиц и заканчивая базами данных различных форматов.

Одной из причин необходимости использования классификаторов является потребность в организации ввода первичных данных. Если пользователю при заполнении полей предоставить возможность выбора из списка предложенных значений, а не заставлять вводить данные с клавиатуры от начала до конца, провоцируя ошибки, можно значительно повысит эффективность работы по заполнению документа, корректность и согласованность полученных данных.

Дружественный интерфейс – «ориентация на человека»

Потенциальные пользователи системы имеют различный уровень квалификации, поэтому пользовательский интерфейс всех уровней системы должен быть удобен для всех категорий пользователей. Интерфейс системы должен упростить решение ежедневных задач и предоставить возможность работы в исключительных ситуациях.

При работе с документами Office CASE должен предоставить экранную форму. Пользователь работает непосредственно только с формой, на стадии заполнения документа у пользователя нет возможности редактировать его напрямую. Разделы экранной формы должны по возможности повторять структуру документа.

Интерактивная проверка введенных пользователем данных также является ключевым требованием к системе. Система должна иметь развитый механизм установки ограничений на конкретные поля документа и информативные контекстные подсказки при вводе некорректных значений полей.

Кроме стандартных элементов управления (метка, текстовое поле, выпадающий список) для организации ввода нестандартных разделов документа потребуются специальные элементы

управления. Например, для ввода повторяющихся фрагментов документа необходим нестандартный элемент управления, который позволит добавлять, редактировать и удалять повторяющиеся фрагменты разделов.

Пользователей системы можно разделить на две основные группы. Первая группа инициирует сбор данных и соответственно является разработчиком форм отчетов. Вторая группа пользователей является «источником данных», то есть заполняет данными предоставленные формы отчетов. Соответственно система Office CASE включает два основных компонента: инструментальные средства, используемые при создании системы документов, и универсальный программный проект документа, реализующий функциональность каждого конкретного документа системы.

Пользователи первой группы непосредственно работают, как с первым так и со вторым компонентом, от них требуются базовые навыки работы с приложениями пакета Microsoft Office и освоение интегрированной среды разработчика систем документов Office CASE. От представителей второй группы требуются только базовые навыки работы с персональным компьютером и программами, имеющими стандартный Windows-интерфейс.

Приоритетным направлением разработки системы является визуализация и упрощение основных операций пользователя при работе с системой.

Мобильность

Для работы с документом не требуется установка никаких дополнительных программных средств, кроме Microsoft Office. Время пользователя не должно теряться на установку и настройку системы. Подготовка документа к работе заключается только в копировании его на жесткий диск конечного пользователя.

Такое требование резко ограничивает круг доступных для использования технологий. Все ключевые компоненты системы в клиентской части приложения необходимо реализовать средствами Visual Basic for Application, минимально используя обращения к системным функциям операционной системы, так как современные антивирусные системы очень жестко подходят к проверке макросов в офисных документах и по умолчанию не позволяют обращаться к каким-либо системным функциям.

Независимость от версии Microsoft Office

Для документов Office CASE требуется обеспечить возможность их использования на платформе Microsoft Office стандартной поставки версии 97 и выше. Объектная модель Microsoft Office и язык Visual Basic for Application постоянно совершенствуются, что делает обеспечение стабильной работы системы нетривиальной задачей.

Следует заметить, что Microsoft Office 97 содержит Visual Basic for Application версии 5.0, начиная с Microsoft Office 2000 версию 6.x. Между двумя версиями языка имеется ряд значительных отличий. Многие важные и полезные функции, которые легко реализуются с 6.x, в 5.0 просто недоступны. Поэтому базовые компоненты, реализующие функциональность документа на стороне пользователя, необходимо писать в терминах Visual Basic for Application версии 5.0 и Microsoft Office 97. При разработке программного проекта документов должны использоваться алгоритмы, работающие одинаково эффективно в различных версиях офисного пакета. Изменения в объектной модели Microsoft Office следует учитывать с помощью механизмов условной компиляции и позднего связывания объектов.

Поддержка языка разметки документов XML

Встроенная поддержка формата XML (eXtensible Markup Language) позволяет интегрировать Office CASE с другими информационными системами.

XML – это универсальный язык разметки, описывающий целый класс объектов данных, называемых XML-документами. Этот язык используется в качестве средства для описания грамматики других языков и контроля правильности составления документов. То есть сам XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания. Например, если нам для обозначения элемента *«Пермь»* в документе необходимо использовать тэг *<city>*, то XML позволяет свободно использовать определяемый пользователем тэг.

После того как пользователь ввел в документ необходимую информацию, он может распечатать документ, поставить подписи руководителей, то есть будет получена привычная «твердая копия» документа. Но для нас этот документ не представляет интереса, наша задача заключается в том, чтобы введенные пользователем данные были получены вышестоящей организацией. Возможно несколько вариантов решения задачи:

- удалить из документа программный проект и метаданные, находящиеся в скрытом разделе;
- сохранить введенные пользователем данные в файле формата XML.

В результате применения первого варианта будет получен обыкновенный документ Word, из которого можно извлечь необходимые данные. К достоинствам этого способа можно отнести то, что во-первых, документ будет представлен в удобном для пользователя виде, во-вторых, при желании можно соблюсти конфиденциальность информации, установив защиту на чтение и редактирование документа. Недостатками этого способа будут сложность и неэффективность извлечения данных, большой объем полученного файла.

Второй вариант основан на том, что XML является оптимальным транспортом для передачи информации. Вместо пере-

дачи всего файла документа можно сгенерировать XML-файл, включающий всю необходимую информацию. Этот файл будет более компактным, может быть обработан не только Office CASE, но и любой программой, способной обрабатывать XML-файлы.

Общий сценарий использования

Office CASE ориентирован на создание иерархически организованной системы документов, предназначенных для сбора первичной информации, представляемой в виде первичных отчетов, и построения сводных отчетов по полученным первичным данным.

Для начала рассмотрим схему функционирования системы документов на конкретном примере, а затем обобщим на произвольный случай.

Рассмотрим работу некоторой организации, имеющей сложную иерархическую организационную структуру. Предположим, что у рассматриваемой организации есть главное управление (Уровень 1), которому подчинены территориальные управления (Уровень 2). Территориальным управлениям, в свою очередь, подчинены филиалы (Уровень 3), непосредственно оказывающие какие-либо услуги.

1. Главному управлению необходимы данные, которые отсутствуют в корпоративной базе данных по причине того, что необходимость в их использовании появилась только сейчас.

В главном управлении создаются шаблоны форм отчетов для филиала и для территориального управления. Затем подготовленные документы рассылаются по территориальным управлениям по электронной почте либо на магнитных носителях или компакт-дисках (рис. А3.1).

2. Получив документы (шаблоны отчетов), территориальные управления рассылают формы отчета по подчиненным им филиалам.

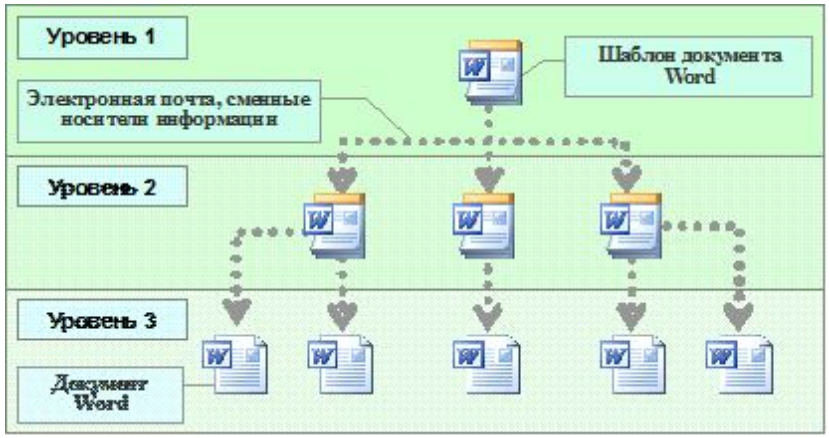


Рис. А3.1. **Распространение форм документов в виде шаблонов Word**

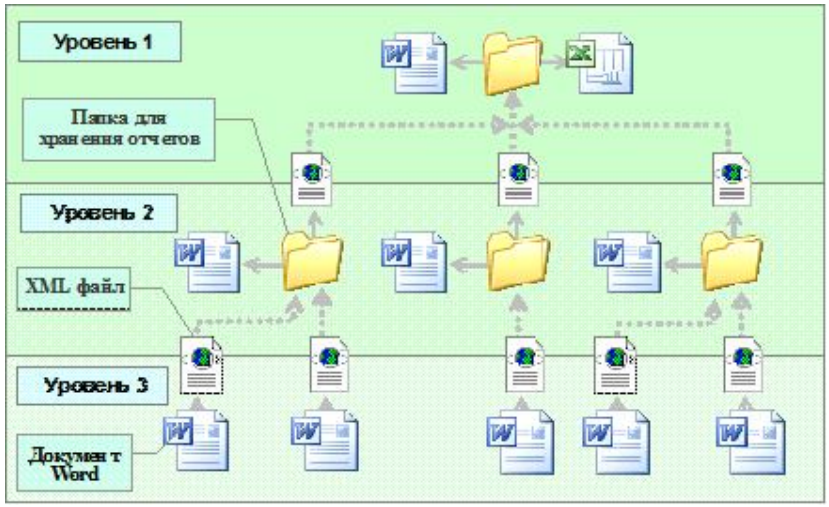


Рис. А3.2. **Сбор данных и формирование сводных отчетов на основе первичных**

3. После получения документа в филиале в него вносится первичная информация, т.е. на основе шаблона создается первичный отчет. При необходимости можно получить твердую копию документа, имеющую юридическую силу. Затем отчет филиала отравляется обратно в территориальное управление в виде документа Word или в формате XML (рис. А3.2).

4. При поступлении в территориальное управление документы сохраняются на жестком диске. При поступлении первичных отчетов всех филиалов на их основе формируется сводный отчет территории, содержащий агрегированные данные, который отправляется в главное управление.

5. В главном управлении, подобно тому как формируется сводный отчет по филиалам в территориальном управлении, формируется сводный отчет по территориальным управлениям. Предусматриваются средства последующей аналитической обработки полученных данных.

Таким образом, мы получили *полностью автоматизированную систему сбора информации*.

В системе присутствуют два типа документов: документы – *контейнеры первичных данных* и документы – *отчеты, агрегирующие данные из первичных документов*. На каждом этапе процесса сбора информации по определенным пользователем правилам данные из первичных документов обрабатываются и попадают в сводный отчет. Количество подобных этапов не ограничено.

Из рассмотренного выше примера можно сделать вывод, что инициатор сбора информации должен последовательно проделать три этапа построения схемы документов. Все остальные функции возьмут на себя компоненты Office CASE.

Первым этапом создания системы документов с помощью Office CASE является *построение схемы системы документов и определение информации, содержащейся в документах*.

Второй этап – *создание визуального представления документа*. Именно с этим представлением документа будет работать пользователь. Microsoft Word располагает широкими возможностями форматирования документа, можно построить форму практически любой сложности.

На третьем этапе Office CASE объединит результаты двух предыдущих. Результатом данного этапа станут *шаблоны, объединяющие в себе структуру документа, программные проекты, реализующие функционирование системы, и метаданные, управляющие поведением документов*. Необязательно, что на данном этапе будут созданы шаблоны всех документов системы. Создаются лишь родительские документы, обладающие базовой информацией.

Структура Office CASE

В системе можно выделить две наиболее важные составляющие: исполнительная среда документа и интегрированная среда разработчика. *Исполнительная среда документа* Office CASE Document Framework (OCDF) – это его программный проект, реализующий всю функциональность на уровне документа. *Интегрированная среда разработчика* Office CASE Documents Studio (OCDS) служит для автоматизированной разработки систем документов. Рассмотрим данные компоненты системы более подробно.

Office CASE Document Framework

Как уже отмечалось ранее, работа пакета Office CASE основана на метаданных. Document Framework, интерпретируя метаданные, содержащиеся в документе, предоставляет пользователю возможность работы на качественно более высоком уровне, чем обычная работа с офисным документом. Например, при заполнении документа первичными данными пользователь вносит данные в экранную форму, причем система в интерактивном ре-

жиме контролирует их корректность; построение сводной документации также может быть организовано в автоматическом режиме, так как в метаданные шаблона сводного отчета включена информация о структуре первичных отчетов.

Документ, созданный в системе Office CASE, состоит из двух основных разделов: скрытого и видимого (рис. А3.3).

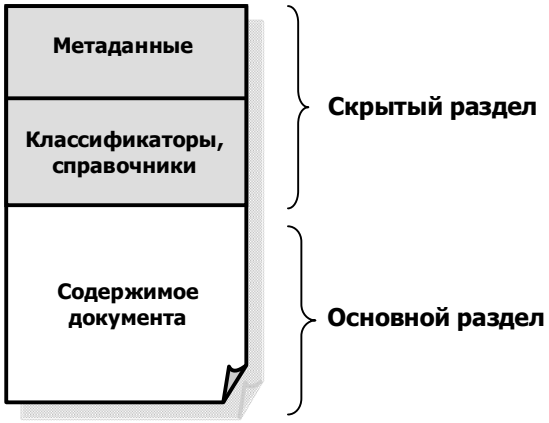


***Рис. А3.3*. Схема Office Case документа**

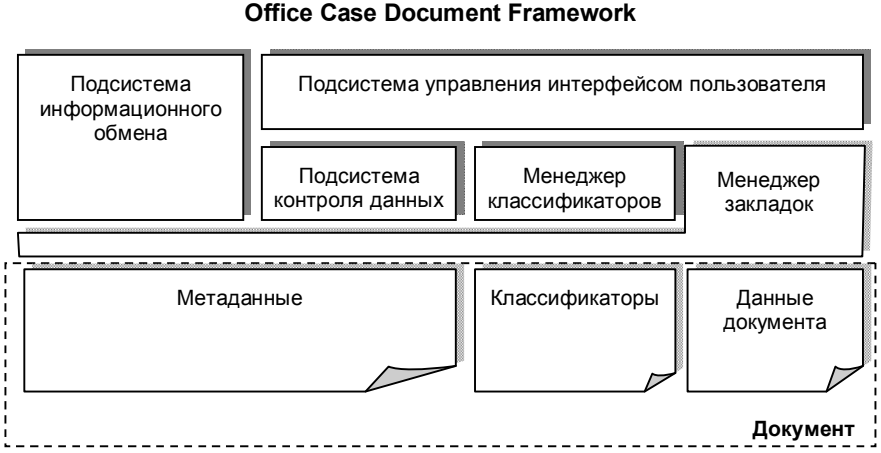


***Рис. А3.4*. Основные подсистемы Office Case Document Framework**

Видимый раздел представляет собой обычный документ: это документ, который в конечном итоге желает получить пользователь. Особенностью раздела является то, что места, в которые должны быть помещены данные, размечены с помощью механизма закладок. Для каждой закладки в метаданных документа указаны правила, по которым может быть получено ее значение. Значение закладки может быть запрошено у пользователя, получено путем обработки известных значений или скопировано из других документов системы.

Основные логические компоненты Document Framework представлены на схеме (рис. А3.4).

Структура скрытого раздела документа

Информация, необходимая для функционирования документа, располагается в скрытом, недоступном пользователю разделе документа.

Скрытый раздел содержит *метаданные*, управляющие поведением документа, и набор *справочников*, используемых при заполнении документа.

Для хранения информации в скрытом разделе используются таблицы. Наиболее важными являются таблицы: закладок, правил, ограничений и описания экранных форм.

Таблица закладок содержит информацию о разметке основного и скрытого разделов документа. В этой таблице хранится информация о типе каждой закладки, способах ее использования. Таблица закладок является одной из самых важных таблиц и имеет всегда фиксированное имя. Доступ ко всем элементам документа, включая таблицы метаданных, осуществляется через закладки.

Информация о правилах, наложенных на вводимые пользователем значения, хранится в *таблице правил* (рис. А3.5). Каждое правило может содержать в себе несколько ограничений. Правило имеет уникальное имя, по которому система к нему обращается. С правилом также связывается текст и тип ошибки, которые выводятся на экран для пользователя, если введенные им данные не удовлетворяют ограничениям правила. Правила и

ограничения находятся в отношении «*многие ко многим*», то есть одно правило может содержать несколько ограничений, и одно ограничение может входить в несколько правил. Для поддержки данного отношения служит *таблица связи правил и ограничений*; поле «Link» данной таблицы используется для связи ограничений в правиле операциями «и» или «или». Информация об ограничениях на значения хранится в *таблице ограничений*. Поле «Name» задает уникальное имя ограничения, по которому к нему обращается система. Поле «Type» используется для указания типа данных проверяемого значения. Поддерживаются типы данных «целый», «вещественный», «строковый» и «дата». Поля «Limit», «Min», «Max» используются для задания допустимого интервала значений проверяемого поля. Булево поле «Empty» указывает, обязательно ли поле для заполнения. Поле «Mask» служит для проверки соответствия проверяемого значения определенному шаблону. Задание поля поле «Format» позволяет отформатировать введенное значении в соответствии с указанным шаблоном.

Id	Name	ErrTitle	ErrMsg	ErrType
1				
2				

а) Таблица правил

Id	IdRule	IdConstraint	Link

б) Таблица связи правил и ограничений

ID	Name	Type	Limit	Min	Max	Empty	Mask	Format

в) Таблица ограничений

***Рис. А3.5*. Структура таблиц хранения правил, налагаемых на вводимую пользователем информацию**

В таблицах информации об экранных формах располагается данные о генерируемых во время работы документа формах для ввода данных. Для хранения данной информации используются три таблицы: *таблица вкладок*, *таблица блоков данных*, *таблица содержания блока данных*.

Формы для ввода данных имеют многостраничную архитектуру, то есть информация логически сгруппирована на вкладках. Для описания вкладок формы используется *таблица вкладок* (рис. А3.6). Каждая из них имеет уникальное имя, по которому система к ней обращается. Поле «Index» используется для управления порядком вывода вкладок. Надпись, выводимая на конкретной вкладке, хранится в поле «Caption».

Id	Name	Index	Caption
1			
2			

***Рис. А3.6*. Структура таблицы хранения информации о вкладках**

Для описания параметров блоков ввода данных используется *таблица блоков данных* (рис. А3.7). Поле «Name» задает уникальное имя блока данных, по которому к нему обращается система. Атрибут «PgName» задает вкладку, на которой будет размещен блок ввода данных. Параметр «Bkmark» указывает на закладку, где располагается *таблица содержания блока данных*. Группа атрибутов «Width», «Height», «Top», «Left» задает расположение блока ввода данных на вкладке. Параметры «FocusR», «FocusC» указывают активный по умолчанию элемент управления.

ID	Name	Use	PgName	Bkmark	Width	Height	Top	Left	FocusR	FocusC
1	dg1	T	0	dgВкл1	476	310	10	10	2	1
2	dg2	T	1	dgВкл2	486	310	10	6	2	1
3	dg3	T	2	dgВкл4	478	130	25	10	2	1
4	dg4	T	2	dgВкл4П	478	140	180	10	2	1
5	dg5	T	3	dgВкл4	474	310	10	10	1	1
6	dg6	T	4	dgВкл5	474	310	10	10	2	1

***Рис. А3.7*. Структура таблицы блоков данных**

Каждый блок данных экранной формы описывается своей *таблицей содержания блока данных*. В ячейках таблицы храниться информация о типе элемента управления, имя закладки, со значением которой связан этот элемент управления, имя правила, которому должно удовлетворять введенное пользователем значение, и другие параметры. В табл. А3.1 и на рис. А3.8 приведен пример таблицы содержания блока данных и сгенерированный на ее основе фрагмент экранной формы.

Таблица А3.1. **Информация о структуре создаваемой экранной формы**

Text="Информация об учреждении"; **MCol**="2"; **ColW**="170"; **Align**="2"; **FontT=**"1";	**Ctrl**="Empty"; **ColW**="285";
Text="Наименование территории:"; **Align**="3";	**Ctrl**="2"; **VR**="Стр"; **BCol**="Name"; **TCol**="Name"; **SrcType**="2"; **DataSrc**="Территории"; **ECol**="Code"; **ExportTo**="ОКАТО"; **CBStyle**="2"; **Bkm**="bmТеррит"; **LinkTo**="2";
Text="Код по ОКАТО:"; **Align**="3";	**Ctrl**="3"; **Bkm**="bmОКАТО"; **LinkTo**="2"; **Name**="ОКАТО"; **Align**="1";
Text="Полное наименование учреждения:"; **Align**="3";	**Ctrl**="1"; **VR**="Стр"; **Bkm**="bmНазвУчреж"; **LinkTo**="2";
Text="Краткое наименование учреждения:"; **Align**="3";	**Ctrl**="1"; **VR**="Стр"; **Bkm**="vrКорНазвОрганизации"; **LinkTo**="3";

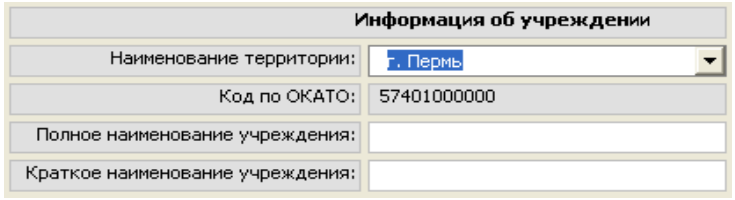


Рис. А3.8. **Фрагмент экранной формы, созданный на основеметаданных, записанных в табл. А3.1**

Менеджер закладок

После того как пользователь заполнил поля экранной формы, введенные данные переносятся в документ, для разметки которого используются стандартные закладки. Использование закладок является наиболее оптимальным подходом при решении задачи разметки документа в общем случае. При заполнении сложного документа трудно точно определить позицию, в которую следует поместить данные, использование закладок дает возможность обращается к нужному диапазону по имени.

Изначально Word и Excel хранят минимум информации о закладке и диапазонах – только ее имя и данные о положении в документе. Для организации полноценной разметки необходима расширенная информация о параметрах закладки: способ получения значения, тип данных и ограничения на хранящееся значение. Вся информация о закладках документа, созданного с помощью Office CASE, как уже отмечалось, хранится в таблице скрытого раздела документа.

Менеджер закладок содержит полный набор процедур для работы с закладками и размеченными диапазонами. Исполнительная среда документа Word и рабочей книги Excel различается лишь данным модулем. Реализация данным модулем некоторого обобщенного интерфейса работы с данными документа позволяет не переписывать другие блоки исполнительной среды.

Менеджер классификаторов

Таблицы классификаторов размещаются в скрытом разделе документа, к которому пользователь не имеет непосредственного доступа. При работе с документом связанные элементы управления экранной формы заполняются значениями. Если пользователь ввел новое значения и такая возможность предусмотрена правилами, наложенными на значение поля документа, то полученное значение сохраняется в таблице, а затем при построении сводного отчета попадет в общий классификатор. Таким способом можно получить новый классификатор и использовать его в создания новых приложений.

В табл. А3.2 представлен фрагмент списка территорий Пермского края. На экранной форме пользователь выбирает территорию по имени, в документ попадает как символьная строка, так и код территории по общероссийскому классификатору административно-территориальных образований. Такой подход позволяет значительно упростить обработку информации и дает возможность интеграции с другими информационными системами.

Таблица А3.2. **Таблица скрытого раздела, служащая источником данных (кодов ОКАТО)**

ID	Code	Name
1	57401365000	Дзержинский район г. Перми
2	57401367000	Индустриальный район г. Перми
3	57401370000	Кировский район г. Перми
4	57401372000	Ленинский район г. Перми
5	57401375000	Мотовилихинский район г. Перми
6	57401378000	Орджоникидзевский район г. Перми
7	57401380000	Свердловский район г. Перми
8	57204000000	Бардымский район
...	...	...
50	57563000000	ЗАТО п. Звездный

Конечно, хранение классификатора непосредственно в документе резко увеличивает его объем и повышает сложность реализации его программного проекта, но преимущества такого подхода компенсируют затраты:

- повышается скорость заполнения документа;
- возможность ввода некорректных данных сводится к минимуму;
- документ существенно легче обрабатывать, так как формат значений стандартизирован;
- документ не зависит от внешнего окружения, то есть наличия или отсутствия драйверов баз данных и других документов.

Подсистема управления интерфейсом пользователя

При работе с документом пользователь не работает непосредственно с «текстом» документа, все данные вводятся с помощью экранной формы, строящейся динамически по метаданным. Примерный вид формы показан на рис.А3.9.

Обычно экранная форма содержит несколько вкладок, которые соответствуют разделам документа. Каждая из вкладок может содержать один или несколько блоков данных.

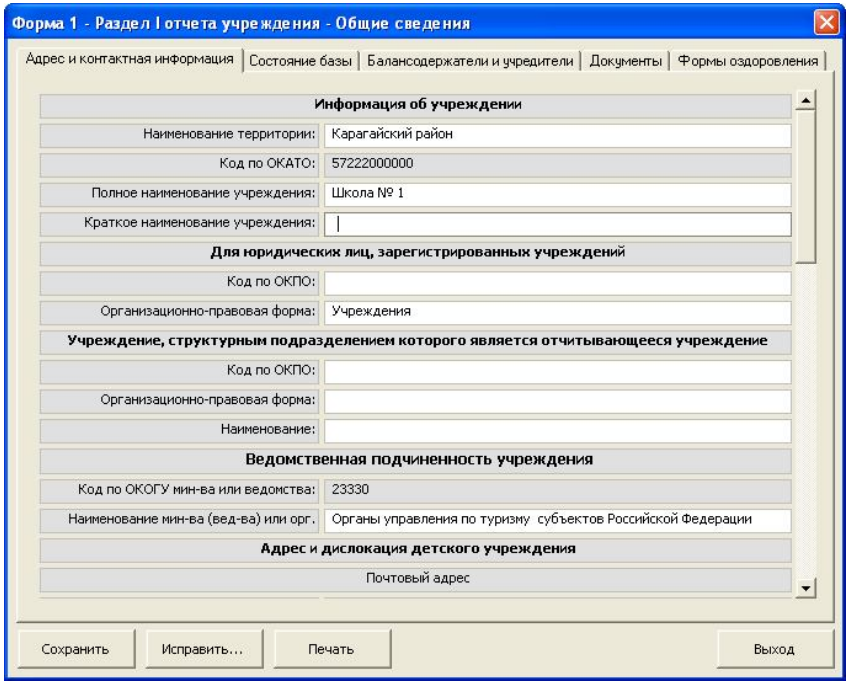


Рис. А3.9. **Пример экранной формы для ввода данных**

Блок данных представляет собой таблицу, в строках и столбцах которой располагаются элементы управления. Есть возможность объединить ячейки таблицы для логической груп-

пировки элементов управления. Для удобства пользователя при большом объеме блока данных применяется механизм прокрутки с возможностью закрепления заголовочных строк и столбцов.

Разработчик документа может произвольным образом настроить внешнее оформление экранной формы, применить различные цветовые схемы оформления.

В любом Office CASE документе, предназначенном для ввода данных, в нижней части формы располагаются стандартные командные кнопки управления документом.

Нажатие на кнопку «Сохранить» приводит к переносу данных с экранной формы в соответствующие поля документа. Пользователь может на некоторое время прервать работу с документом. При последующей загрузке документа данные из документа загрузятся в экранную форму.

Кнопка «Исправить» приводит к запуску проверки корректности данных в экранной форме. Помимо интерактивной проверки данных непосредственно в процессе набора данных такой механизм бывает весьма полезен.

В процессе работы с документом есть возможность переключиться с представления экранной формы на представление документа. Но у пользователя нет возможности модифицировать документ.

В качестве элементов управления экранной формы могут выступать: текстовое поле, надпись, комбинированный список и счетчик. Информация для создания экранной формы загружается из таблицы, находящейся в скрытом разделе документа.

Подсистема контроля данных

Интерактивный контроль данных при заполнении полей документа является очень важной функцией в рассматриваемой системе. При запуске документа информация о доступных правилах и ограничениях загружается из таблицы правил. При изменении значения поля введенное значение проверяется на соответствие ограничениям.

Подсистема информационного обмена

Основная задача рассматриваемой подсистемы – это представить интерфейс и набор функций для извлечения информации из документа. Эта подсистема также отвечает за генерацию XML-файла по данным документа.

Выше мы рассматривали документы, являющиеся первичными, то есть документы, содержащие данные, которые вводит пользователь. Office CASE позволяет также строить документы, агрегирующие первичные данные: документ формируется на базе заполненных документов.

Программный проект документа рассматриваемого типа должен реализовать две функции: *создание документов*, в которые будет введена первичная информация, и *обработка информации* из заполненных документов.

При создании дочерних документов документ-родитель сохраняет о них информацию, которую сможет использовать при создании отчета (например, не допустить повторной обработки первичного отчета или указать, какого первичного документа не хватает для создания сводного отчета).

Для реализации второй функции служит специальный документ-мастер, последовательно собирающий необходимую информацию из первичных документов. Результат работы Мастера – готовый документ, содержащий агрегированную информацию.

На рис. А3.10 показано окно Мастера, создающего сводный отчет государственной статистической отчетности по форме «76-рик» на основе первичных отчетов по форме «ОШ-1» образовательных учреждений.

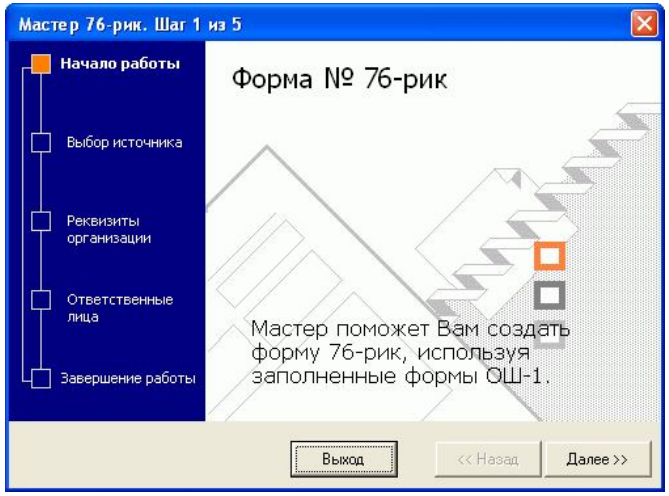


***Рис. А3.10*. Первый этап работы мастера создания формы 76-рик**

Office CASE Documents Studio

Основная задача Office CASE Documents Studio (OCDS) – автоматизация разработки систем документов. В интегрированной среде пользователю предоставляется возможность полного цикла разработки: от проектирования структуры документов до описания схемы консолидации и обработки данных. Особенностью системы является ее ориентация на пользователя-непрограммиста, поэтому отдается предпочтение визуальным инструментам. Разработчик получает возможность манипуляции объектами Office CASE Document Framework и объектами самого Microsoft Office без какого-либо программирования на VBA.

Documents Studio представляет собой отдельное приложение, с которым будет работать только администратор, ответственный за создание шаблонов автоматизированных документов. Программа устанавливается только на рабочем месте администратора. Рядовому пользователю, работающему с документами,

разработанными с помощью Office CASE, навыки работы в Office CASE Documents Studio не нужны.

Рассмотрим более подробно архитектуру разрабатываемых с помощью Office CASE систем документов. Система состоит из набора связанных по данным документов. Документ может состоять из нескольких разделов. Каждому разделу документа ставится в соответствие один или несколько документов программы Microsoft Office (документ Word, рабочая книга Excel). При наличии в документе нескольких разделов доступ к ним осуществляется только через главный документ. Главный документ формы управляет созданием документов разделов из шаблона, последовательностью заполнения, тиражированием информации между разделами, подготовкой отчета для передачи. В тривиальном случае документ состоит из единственного раздела и не содержит главного документа.

Для создания системы документов с помощью Office CASE Documents Studio пользователю необходимо пройти четыре этапа:

1. *Разработка внешнего представления документов*, разметка полей для ввода данных – это задача первого этапа.
2. *Построение диаграммы документооборота* является вторым этапом. Пользователь должен указать, какие объекты участвуют в процессе и как они связаны между собой.
3. После создания диаграммы следует этап *уточнения схемы – задание свойств объектов*. На данном этапе пользователь настраивает экранные формы для ввода данных.
4. На конечном этапе работы Documents Studio автоматически *сгенерирует шаблоны документов системы*. В шаблоны будут импортированы необходимые классы и модули Office Case Document Framework, внедрены метаданные, управляющие поведением документа. После данного этапа документы системы могут начать свое автономное функционирование согласно схеме, предусмотренной разработчиком.

Основные подсистемы Office CASE Documents Studio представлены на рис. А3.11.

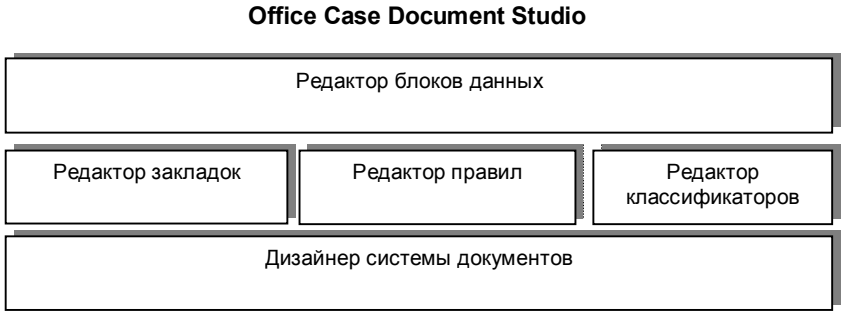


***Рис. А3.11*. Основные подсистемы Office Case Document Studio**

Дизайнер системы документов

Данный компонент предназначен для описания схемы взаимодействия элементов документооборота. С помощью визуальных средств пользователь создает и настраивает свойства элементов документооборота. На основе разработанной схемы генерируются все документы системы. Схема является основой для генерации программного проекта документов.

Интерфейс пользователя компонента представлен на рис. А3.12.

Схема документооборота представлена семантической сетью. Вершины сети соответствуют объектам документооборота. Дуги используются для представления операций над объектами. Например, шаблон и документ, созданный на основе шаблона, должны быть соединены дугой «Создать».

Основные объекты схемы документооборота представлены в табл. А3.3.

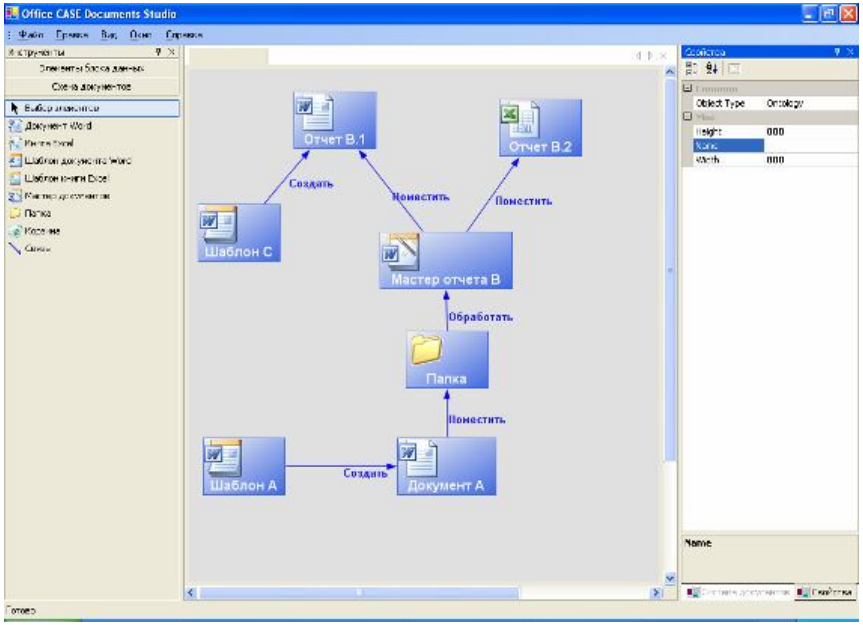


***Рис. А3.12*. Компонент описания системы документов**

Редактор классификаторов

Компонент предназначен для работы с классификаторами, используемыми в системе документов. Имеется специальный компонент для импорта существующих классификаторов из различных источников. Операция выполняется с помощью средств Мастера импорта данных. При этом появляется окно Мастера, показанное на рис. А3.13. Редактор позволяет вносить изменения в классификаторы (рис. А314).

Редактор правил

Компонент предназначен для создания правил и ограничений, налагаемых на данные документа.

Каждое правило может содержать в себе несколько ограничений. Правило имеет уникальное имя, по которому система к нему обращается. Также с правилом связывается текст и тип ошибки, выводящиеся пользователю на экран, если введенные

им данные не удовлетворяют ограничениям правила. Правила и ограничения находятся в отношении «*многие ко многим*», то есть одно правило может содержать несколько ограничений, и одно ограничение может входить в несколько правил.

Таблица А3.3. **Основные объекты системы документов**

Пикторгамма	Название объекта	Описание
	Документ Word	Один из основных объектов схемы. Представляет собой документ Microsoft Word с программным проектом и внедренными метаданными. После компиляции документ представляет офисное приложение, с которым работает пользователь
	Шаблон документа Word	Шаблон используется для создания документов Word
	Рабочая книга Excel	Один из основных объектов схемы. Представляет собой документ – рабочую книгу Excel с программным проектом и внедренными метаданными. После компиляции документ представляет офисное приложение, с которым работает пользователь
	Шаблон рабочей книги Excel	Шаблон используется для создания рабочей книги Excel
	Мастер документов	Объект используется для создания сводных отчетов на базе первичных
	Консолидация данных	Объект используется для построения сводных отчетов: консолидации (агрегирования) данных, полученных из первичных отчетов
	Копирование данных	Объект используется для построения сводных отчетов: переноса данных первичных отчетов в таблицы сводных отчетов
	Папка документов	Объект используется для сбора и хранения документов

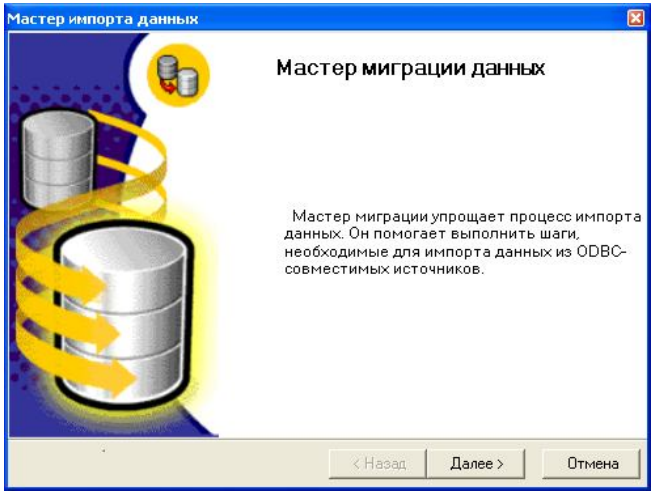


Рис. А3.13. **Мастер миграции данных**

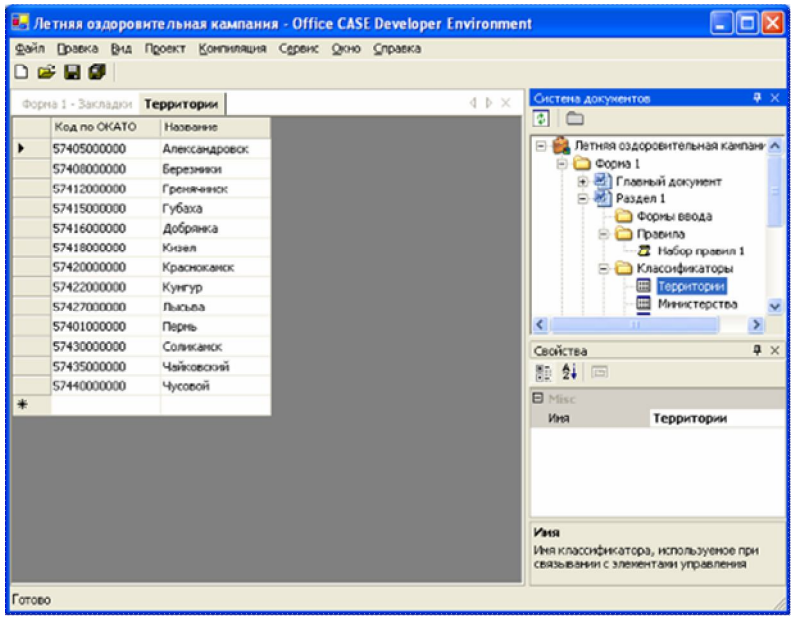


Рис. А3.14. **Редактор классификаторов Office Case Document Studio**

С правилом связывается тип данных, которому должно принадлежать проверяемое значение. Маска ввода служит для форматирования значения, если оно удовлетворяет данному правилу.

С каждым ограничениям правила могут быть связаны свои реакция на ошибку и текст, выдаваемый пользователю. Само ограничение задается как отношение между двумя значениями.

Редактор закладок

Компонент предназначен для упрощения разметки документа. В интегрированную среду включен непосредственно документ Word, в котором пользователь в визуальном режиме включает закладки.

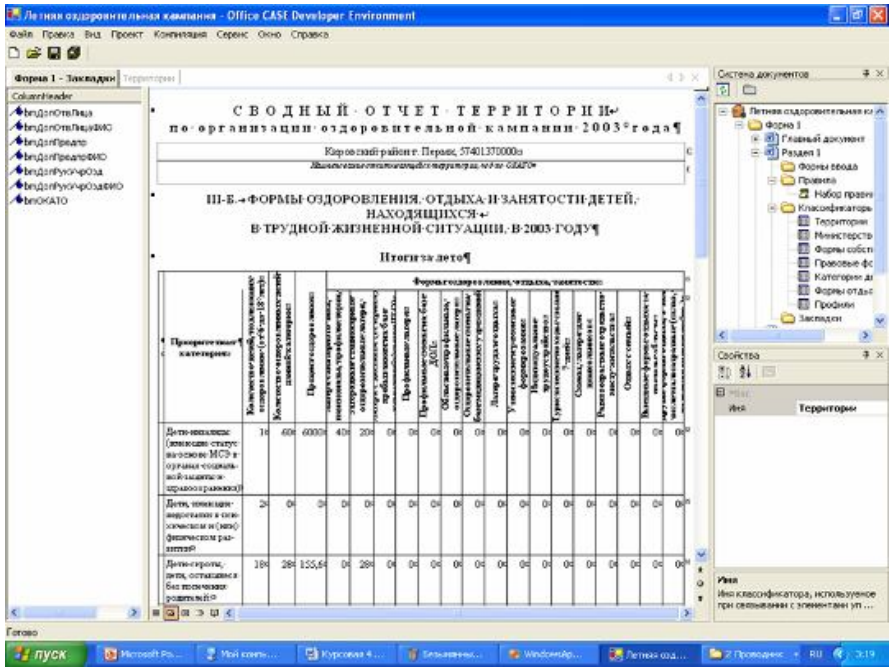


Рис. А3.15. **Редактор закладок Office Case Document Studio**

Редактор блоков данных

Редактор блоков данных – это инструмент, позволяющий в визуальном режиме разрабатывать блоки данных экранных форм, используя разметку таблицы.

В табл. А3.4 перечислены доступные для вставки в блок данных элементы управления.

Таблица А3.4. **Доступные для вставки в блок данных элементы управления**

	Название	Назначение	Основные свойства
	Пустая ячейка	Используется для форматирования таблицы	Цвет фона
A	Надпись	Используется для вывода текстовой надписи я ячейке таблице	Цвет фона Цвет текста Текст Выравнивание текста
abl	Текстовое поле	Ввод и редактирование данных	Значение Правило Закладка Выравнивание текста
	Выпадающий список	Выбор данных из списка	Значение Правило Закладка Источник данных

Для внесения и изменения разметки необходимо выделять ячейки таблицы (рис. А3.16-А3.19).

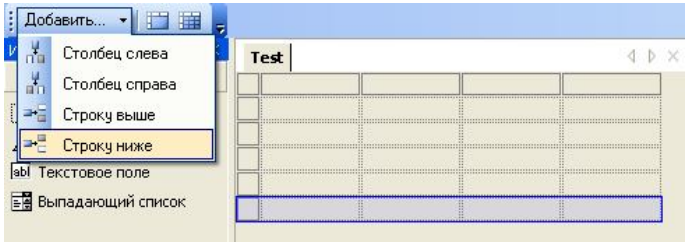


Рис. А3.16. **Добавление строки в таблицу разметки**

Для выделения отдельной ячейки необходимо щелкнуть по ней мышью. Для выделения нескольких ячеек необходимо, зажав клавишу *Shift* щелкнуть требуемые ячейки. Выделение всех

ячеек стрики или столбца можно осуществить с помощью двойного щелчка по заголовку строки или столбца.

Изменения применяются ко всем выделенным элементам управления. При выделении разнотипных элементов управления в окне «Свойства» отображаются только общие свойства.

Для изменения ширины столбца необходимо выделить его щелчком мыши и установить требуемое значение параметра «Ширина». (рис. 57) Для изменения высоты строки необходимо выделить ее щелчком мыши и установить требуемое значение параметра «Высота». При изменении размеров строк (рис. 58) и столбцов изменяются размеры находящихся в них элементов управления. В окне «Свойства» могут быть изменены и другие параметры свойств и столбцов.

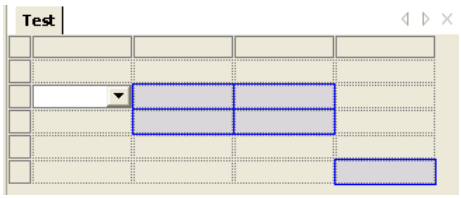


***Рис. А3.17*. Выделение ячеек**

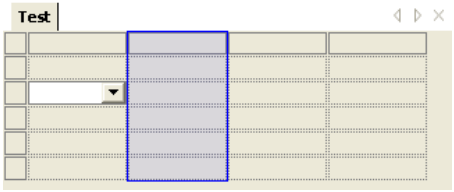


***Рис. А3.18*. Выделение столбца**

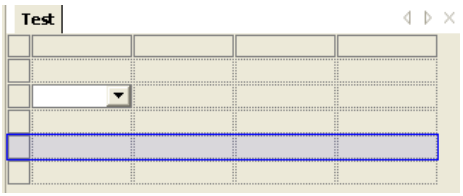


***Рис. А3.19*. Выделение строки**

Для изменения типа элемента управления необходимо выделить ячейки таблицы разметки и установить требуемый тип элемента управления (рис. А3.20). По умолчанию в ячейках таблицы расположен элемент управления «Пустая ячейка».

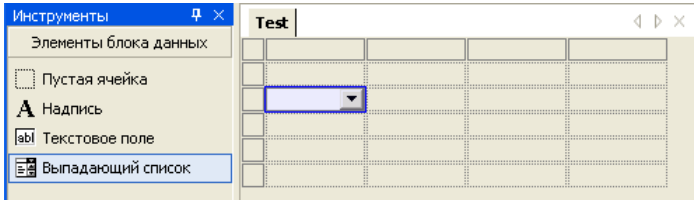


***Рис. А3.20*. Размещение элемента управления**

Смежные ячейки таблицы разметки могут быть объединены. Расположенный в объединенной ячейке элемент управления займет все пространство ячейки (рис. А3.21).

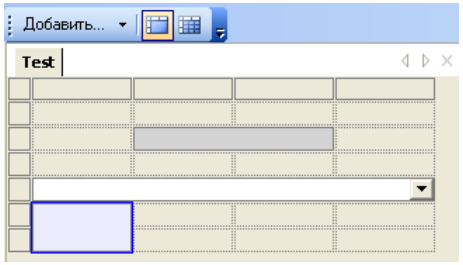


***Рис.А3.21*. Объединения ячеек таблицы разметки**

Для объединения ячеек необходимо выделить смежные ячейки и нажать на кнопку «Объединить ячейки» панели инструментов «Таблица разметки». Для удаления объединения ячеек надо нажать на кнопку «Разбить ячейки» панели инструментов «Таблица разметки».

Пример применения Office CASE

Постановка задачи

В Пермской области по инициативе Отдела по защите прав детей администрации области была создана межведомственная региональная система учета отдыха, оздоровления и занятости детей.

Ранее при планировании и подведении итогов оздоровительной кампании возникали проблемы, связанные с недостатком информации для рационального распределения средств, двойным учетом детей, проходящих по спискам различных ведомств. Достоверный анализ получаемой информации был затруднен вследствие ее большого объема и невозможности автоматизации обработки.

Решением проблемы является персонифицированный учет детей. Для его реализации было предложено создать информационную систему, охватывающую учреждения образования, здравоохранения, социальной защиты, службы занятости, подразделения ОВД и КДН, комитеты по делам молодежи, комитеты по физической культуре и спорту, управления культуры и искусства и др.

Создание информационной системы регионального масштаба, которая позволила бы вести оперативный учет и мониторинг, требовало значительных ресурсов. На первом этапе для решения поставленной задачи была разработана система взаимосвязанных документов, представляющих первичные и сводные отчеты учреждений и территорий области. Данные, получаемые из отчетов, представлены в форме, пригодной для передачи их в базу данных информационной системы и ее первичного наполнения (рис. А3.22).

Разработанная система дает возможность хранить информацию о состоянии здоровья ребенка, основные сведения о его семье, материальном положении и условиях жизни, о постановке на учет в КДН и ПДН. Эта информация является основой для планирования отдыха, оздоровления и занятости детей приоритетных категорий. В базе данных представлена информация об учреждениях оздоровления. Для каждого учреждения хранится информация о его материальной базе, реализуемых формах отдыха, стоимости путевок и т.д. Система позволяет учитывать использование финансовых средств, получаемых из различных источников.

Программное обеспечение ИС позволяет создавать документы в соответствии с утвержденными формами, передавать данные по запросам пользователей в электронные таблицы для их аналитической обработки.

С целью обеспечения сбора информации о результатах летней оздоровительной кампании в виде, пригодном для дальнейшего анализа информации с помощью компьютера, а также получения данных для первичного наполнения системы разработано специальное программное обеспечение – «Подсистема автоматизации подготовки отчетов», позволяющая снизить трудоемкость формирования отчетов в соответствии с разработанными едиными формами отчетов. «Подсистема автоматизации подготовки отчетов» реализована с помощью Office CASE.

Система документов включает три уровня отчетности: отчеты учреждений оздоровления и структурных подразделений администраций районов, отчеты территориальных координационных советов и отчеты регионального уровня. Учреждения, участвующие в оздоровительной кампании, различаются по уровню обеспеченности вычислительной техникой, средствам передачи информации. Их специалисты имеют различные уровни подготовки.

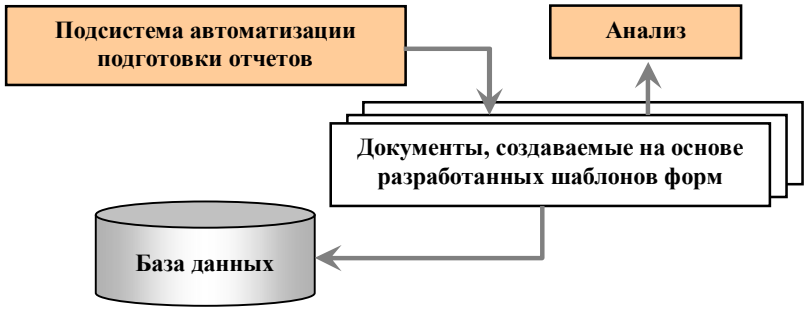


Рис. А3.22. **Схема интеграции «Подсистемы автоматизации подготовки отчетов»**

Разработанное программное обеспечение минимизирует возможность внесения ошибок в отчетные данные, обеспечивает их согласованность и возможность интеграции при формировании сводных отчетов.

Данные вводятся в отчетные формы в виде, пригодном для дальнейшей компьютерной обработки и переноса полученных данных в создаваемую базу данных для ее первичного наполнения и передачи пользователям при внедрении системы.

Реализация системы документов

Документы «Подсистемы автоматизации подготовки отчетов»

Отчеты о проведении оздоровительной кампании представляются в электронном виде, предполагающем возможность их дальнейшей машинной обработки, и на бланках специальных форм. Формы отчетов являются унифицированными, едиными для всех территорий. Перечень отчетов включает следующие формы:

- *Форма 1* – отчет учреждения, организующего оздоровление детей в возрасте от 6 до 18 лет на территории района (города).
- *Форма 2* – отчет структурного подразделения администрации района (города) по персонифицированному учету оздоровления детей в возрасте от 6 до 18 лет.
- *Форма 3* – сводный отчет территории, интегрирующий данные, полученные из первичных отчетов учреждений и структурных подразделений администрации территории.
- *Форма 4* – текстовой отчет территории.

Формы отчетов разбиты на разделы, включающие как первичные данные, характеризующие деятельность учреждений, организаций, так и результирующие (итоговые) данные, предназначенные для проведения анализа и обобщения, вычисляемые на основе введенных первичных данных.

Форма 1 разбита на несколько разделов, характеризующих различные аспекты деятельности учреждений. Каждый раздел

отчета может заполняться и предоставляться в координационный совет территории или вышестоящие органы управления отдельно. Ниже приведен перечень разделов отчета с указанием, учреждениями каких типов эти разделы должны заполняться:

Раздел I – *Общие сведения* – заполняется *всеми учреждениями*. Информация, включенная в этот раздел, используется при формировании других разделов отчета и отчетов, создаваемых в соответствии с Формами 2 (используется список учреждений оздоровления при вводе персональных данных о распределении путевок в оздоровительные учреждения) и 3 (формируется общий список всех учреждений оздоровления территории, включающий сведения об учредителях, балансодержателях и арендодателях учреждения).

Раздел II – *Затраты на содержание учреждения и источники его финансирования в 2003 году* – заполняется *всеми учреждениями*. Данные, включенные в этот раздел отчета, используются при формировании отчетов, создаваемых в соответствии с Формами 2 (при формировании персонифицированных сведений об оздоровлении детей используются данные о реализованных учреждением путевках) и 3 (формируются данные о деятельности учреждения в летний период, включающие сведения об источниках его финансирования).

Раздел III – *Сведения об оздоравливаемых детях* – заполняется *всеми учреждениями*. Включает сведения о детях, отдыхавших в отчитывающемся учреждении летом 2003 года. На основе данных персонифицированного учета формируются сводные таблицы отчета учреждения (Раздел III-Б). Кроме того, список детей используется для формирования показателей эффективности оздоровления и эффективности работы учреждения (Раздел V). Данные о детях переносятся в сводный отчет территории и на их основе формируются результатные показатели сводного отчета координационного совета территории.

Раздел IV – *Сведения об учащихся и воспитанниках* – заполняется только образовательными учреждениями общего образования, начального профессионального образования, детскими домами и школами-интернатами и т.п.

Раздел V – *Показатели эффективности работы оздоровительного учреждения*. *Всеми учреждениями* оздоровления фор-

мируются следующие подразделы этого раздела отчета: «Обращения за медицинской помощью», «Информация о детях, покинувших оздоровительное учреждение до завершения смены», «Сведения о проведенных спортивно-оздоровительных и культурно-массовых мероприятиях» (на основании этих сведений оценивается эффективность деятельности учреждения при формировании сводного и текстового отчета территорий).

Раздел VI – *Материально-хозяйственная база учреждения* – формируется *в обязательном порядке только загородными оздоровительными лагерями* (сведения этого раздела необходимы для проведения инвентаризации загородных баз и оценки эффективности их использования). При формировании данного раздела отчета заполнение ведется с учетом специфики учреждений различных типов, их деятельности и используемой ими материальной базы. Включенная в этот раздел информация используется для анализа состояния имеющейся базы и оценки эффективности мер по сохранению загородных баз, которые должны быть представлены в текстовом отчете территории.

Раздел VII – *Кадровое обеспечение учреждения*. При формировании данного раздела отчета заполнение ведется с учетом специфики учреждений различных типов, их деятельности и потребностей в использовании квалифицированных кадров. Включенная в этот раздел информация используется для анализа проблем, оценки использования новых форм работы с кадрами и пр. при формировании текстового отчета территории.

Раздел VIII – *Предложения по дальнейшему использованию базы учреждения* – в обязательном порядке сведения о перепрофилировании и передаче базы учреждения в муниципальную собственность заполняются только загородными оздоровительными лагерями. Предложения по совершенствованию работы учреждения оздоровления заполняются всеми учреждениями.

Отчитывающиеся учреждения в первую очередь должны подготовить информацию Разделов I и II и представить ее в координационные советы территорий для формирования *общего списка учреждений оздоровления*, который должен использоваться при создании отчетов в соответствии с Формами 2 и 3. Наличие и использование общего списка при подготовке отчетов позволит интегрировать данные, полученные из различных источников. Сведения об организации отдыха всех детей, являющихся учащимися или воспитанниками образовательных учреждений (школ, профтехучилищ, детских домов и т.п.), включенные в Раздел IV, могут быть сформированы только после предоставления координационным советом списка всех оздоровительных учреждений, работающих на данной территории.

Отчеты по *Форме 2* могут быть подготовлены структурными подразделениями после подготовки координационным советом территории общего списка всех учреждений оздоровления (учреждения, включенные в этот список, являются местами отдыха детей, в них включается информация о реализованных формах отдыха и профилях проведенных смен, стоимости путевок, что необходимо знать при формировании результирующих данных по итогам оздоровительной кампании).

Сводный отчет территории (Форма 3) и текстовый отчет координационного совета (Форма 4) формируются на основе данных, включенных в отчеты учреждений и структурных подразделений (Формы 1 и 2).

Таким образом, можно установить следующий порядок формирования отчетов (рис. А3.23):

– Заполнение Разделов I и II Формы 1 всеми учреждениями оздоровления и передача их в координационные советы территорий для создания общих списков учреждений (для формирования Раздела 2 «Сведения об учреждениях оздоровления…» сводного отчета по Форме 3).

– Формирование координационными советами общих списков оздоровительных учреждений на основе полученных от каждого учреждения данных и передача сформированных списков отчитывающимся учреждениям и структурным подразделениям.

– Формирование отчетов по Формам 1 (оставшиеся разделы отчета) и 2 учреждениями и структурными подразделениями и передача их в координационные советы и вышестоящие органы управления для обобщения и анализа.

– Подготовка сводного отчета территории (оставшиеся разделы в соответствии с Формой 3) и текстового отчета координационного совета (Форма 4).

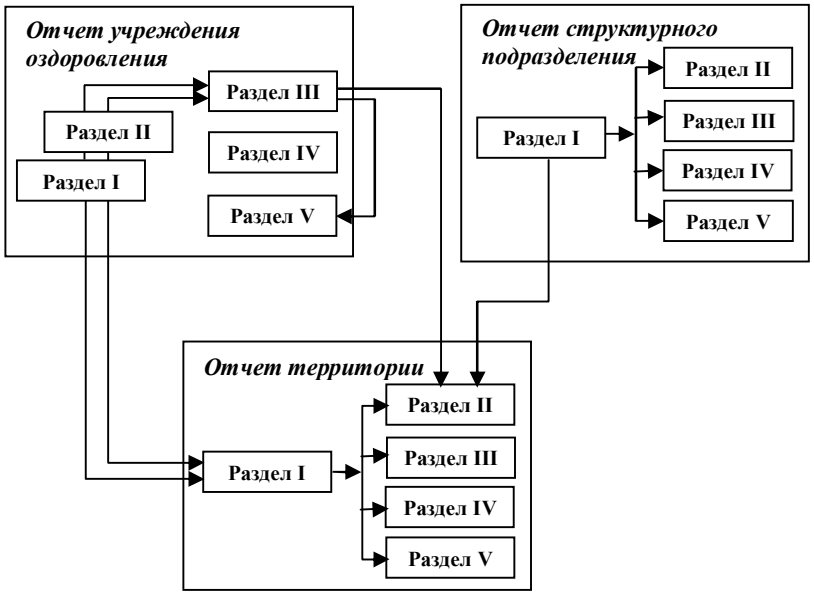


***Рис. А3.23*. Схема формирования отчетов (связи между документами)**

При формировании отчетов информация, полученная из отчетов одной формы, может переноситься в отчеты, создаваемые по другим формам. Для обеспечения возможности обмена данными и их последующей компьютерной обработки требуется соблюдать единообразие представления данных при формировании отчетов:

- при вводе данных можно использовать только те обозначения, которые предлагаются соответствующей формой или инструкцией по ее заполнению;
- нельзя изменять структуру отчета (удалять и перемещать его подразделы, включенные в отчет таблицы, удалять столбцы таблиц (можно добавлять только строки), переименовывать графы (столбцы) и строки таблиц и т.п.).

Формат ввода данных приведен в инструкциях по заполнению отчетов и руководствах пользователей подсистемы автоматизации подготовки отчетов, а также в подсказках, отображенных в диалоговых окнах, предназначенных для ввода данных.

Требования к квалификации пользователей

Для установки и использования системы автоматизации подготовки отчетов специальных требований к уровню и квалификации пользователей не предъявляется.

Пользователи должны владеть базовыми навыками работы в среде операционной системы (ОС) Windows (9x/ME или NT/2000/XP/2003):

- знание основных элементов интерфейса пользователя, владение базовыми приемами работы (операции файловой системы (создания, копирования перемещения файлов и папок), ввода и редактирования текстов);
- умение работать с приложениями электронной почты.

Приложение полностью основывается на использовании интегрированного пакета Microsoft Office. Для подготовки шаблонов отчетов и формирования документов на их основе применяется текстовый процессор Microsoft Word. Для промежуточного хранения введенных данных и формирования сводных отчетов используются электронные таблицы Excel. Кроме того, с помощью Excel визуализируются в виде диаграмм результаты обработки данных, которые могут быть выведены в качестве приложений к отчету для иллюстрации представленных в нем данных. Пользователи, устанавливающие пакет подготовки отчетов, должны уметь не только работать в среде Microsoft Office, но и делать необходимые установки его параметров, а также параметров ОС. После установки приложения вводить данные и получать результаты работы в виде отчетов может любой пользователь, знакомый с работой в среде Word (требуются навыки ввода и редактирования текста, работы с элементами управления в диалоговых окнах, умение выбирать устройства и папки для сохранения данных, настраивать параметры печати).

Установка приложения и подготовка к работе

Приложение полностью основано на использовании возможностей стандартного выпуска Microsoft Office (97/2000/XP/2003) со следующими установленными приложениями и средствами пакета:

- текстовый процессор Word,

– электронные таблицы Excel,
– библиотеки и язык программирования пакета VBA (Visual Basic for Applications).

Приложение не требует установки какого-либо дополнительного программного обеспечения.

Все функции по подготовке отчетов (ввод первичных данных и формирование результирующих данных в отчетах, генерация сводных отчетов) внедрены в документы и шаблоны Word и рабочие книги Excel.

Созданные шаблоны можно передать любым способом. Файлы можно упаковоть с помощью архиваторов.

Подготовленные с помощью подсистемы данные отвечают правилам представления и кодирования информации в базе данных межведомственной информационной системы. Введенная информация может быть использована не только для подготовки отчетов и анализа, но и для последующего первичного наполнения базы данных и подготовки ее для работы по планированию, учету отдыха, оздоровления и занятости детей.

Для подготовки системы к работе необходимо выполнить следующие действия:

1. Создать на жестком диске компьютера папку (например, «Отчет учреждения оздоровления»).

2. Скопировать саморазворачивающийся архивный файл «Форма 1. Отчет учреждения.exe» в рабочую папку на жесткий диск компьютера (или сохранить вложение, содержащее этот архивный файл, если он получен по почте, в рабочую папку).

3. Запустить программу разархивации архивных файлов двойным щелчком мыши по значку файла саморазворачивающегося архива.

4. В поле ввода «Папка назначения:» установить созданную папку «Отчет учреждения оздоровления». (Для выбора этой папки можно воспользоваться кнопкой «Обзор…».)

5. После выбора нужной папки щелкнуть по кнопке «Извлечь». Ход извлечения файлов из архива отображается в диалоговом окне программы.

6. После закрытия программы разархивации открыть папку «Форма 1», вложенную в созданную папку. В ней должны быть

размещены используемые подсистемой подготовки отчетов файлы и папки:

– файл «Форма 1.doc» – файл, позволяющий выбрать раздел отчета для просмотра или редактирования (изменения);
– DataTables – папка, в которой в формате таблиц размещаются данные, используемые для подготовки разделов отчета учреждения оздоровления по Форме 1;
– Sections – папка, в которую помещаются после генерации разделы отчета учреждения в формате документов Word (каждый раздел отчета представляет собой документ Word);
– Templates – папка, в которой размещаются шаблоны Word разделов отчета учреждения, на которые разбит отчет по Форме 1 (каждому разделу соответствует собственный шаблон);
– Инструкции – папка, в которую помещаются инструкции по подготовке отчетов;
– Готовый отчет – папка, в которую приложение помещает подготовленный для передачи в вышестоящие организации отчет (включаются все необходимые для его обработки и создания сводных отчетов файлы).

Если все перечисленные папки и файлы размещены в созданной папке, приложение готово для работы.

Приложения системы автоматизации подготовки отчетов основаны на использовании макросов, написанных на языке программирования MS Office VBA, поэтому перед запуском приложения автоматизации подготовки отчетов нужно отключить защиту от макровирусов в документах Word и Excel (MS Word используется для разметки форм отчетов и их заполнения; MS Excel используется для промежуточного хранения данных и их обработки при формировании сводных отчетов и визуализации их результатов). При отключенной защите запрос об отключении макросов не будет выводиться на экран. После завершения работы с программой лучше снова включить защиту, чтобы подстраховаться от заражения вирусами через другие

документы, полученные из различных, возможно, непроверенных, источников.

Интерфейс первичных документов

Для запуска приложения, позволяющего создать и отредактировать разделы документа, необходимо открыть файл (документ Word). После открытия документа (запуска приложения) на экране появляется форма (рис. А3.24), позволяющая выбрать раздел отчета для выполнения над ним операций редактирования, подготовки к отправке, печати.

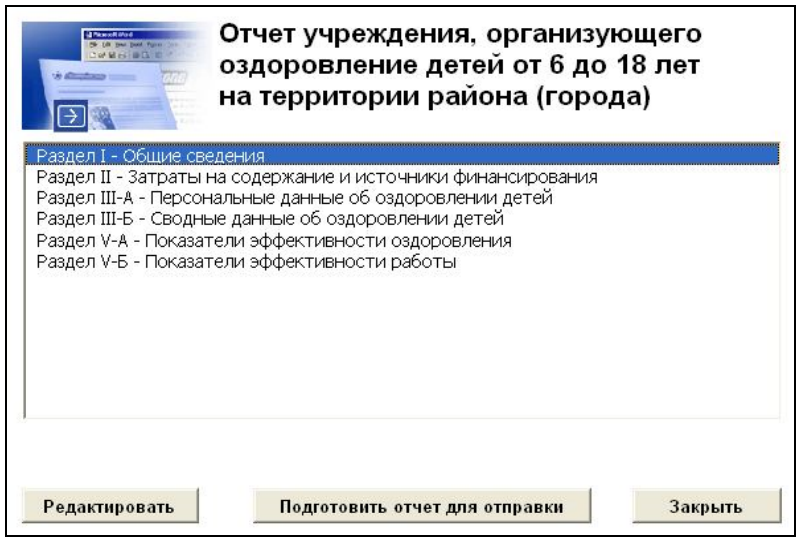


***Рис. А3.24*. Главный документ формы, позволяющий получить доступ к разделам**

Для начала работы следует выбрать нужный раздел отчета, щелкнув по его названию мышью (соответствующая строка окажется выделенной), и затем щелчком мыши по командной кнопке выполнить нужную команду.

Кнопка «Редактировать» открывает диалоговое окно, в котором можно просмотреть включенные в выбранный раздел отчета данные, изменить их, вывести раздел на печать. Если выбранный раздел отчета еще не существует, щелчок по кнопке

«Редактировать» создает документ Word в соответствии с шаблоном этого раздела и готовит все вспомогательные данные для его формирования. Если раздел отчета был создан ранее, раскрывается диалоговое окно, в котором отображаются уже внесенные в него данные. Разделы отчета формируются в соответствии с описанным порядком. Если просмотр информации и ее редактирование возможны, запускается приложение, предназначенное для подготовки выбранного раздела отчета. Его запуск может потребовать времени, если в отчет уже включены большие массивы информации. Ход загрузки данных отображается в окне (рис. А3.25).

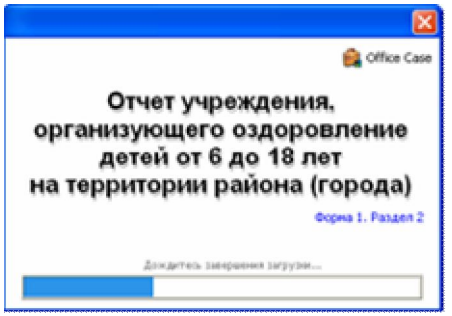


***Рис. А3.25*. Стартовое окно документа**

Далее открывается диалоговое окно, в котором отображаются внесенные в отчет данные, содержимое отчета можно менять, используя элементы управления, расположенные в окне.

Кнопка «Подготовить отчет для отправки» готовит все данные, используемые для формирования отчета и необходимые для генерации сводных отчетов.

Кнопка «Закрыть» завершает работу с приложением.

В приложении реализован стандартный интерфейс Windows. Для выбора действий, ввода и редактирования первичных данных, сохранения результатов работы используются стандартные элементы управления Windows: кнопки, флажки и переключатели, поля ввода и поля ввода со списками.

Информация в экранной форме размещается на вкладках, соответствующих подразделам разделов отчетов. Размещение

элементов управления и их вид при редактировании максимально приближены к виду печатной формы документа.

При вводе первичных данных контролируется правильность ввода (на уровне контроля типов вводимых данных (даты, числа) и допустимых диапазонов значений).

При разработке приложения реализован принцип «не набирай, а выбирай»: все значения данных, которые определены нормативными документами и повторяются в различных разделах отчета или в отчетах различных организаций, используются для сопоставления данных и их анализа (списки территорий Пермской области, форм собственности, организационно-правовых форм, категорий детей, находящихся в трудной жизненной ситуации, форм отдыха и пр.), определены в приложении и выбираются пользователем при вводе данных из раскрывающихся списков. В тех случаях, когда эти списки могут расширяться (например, список учреждений оздоровления), пользователю предоставляется возможность ввода новых значений.

Если вся информация выбранного подраздела не может быть одновременно отображена в окне, на вкладке появляются вертикальная и/или горизонтальная полосы прокрутки.

Между элементами управления в окне можно перемещаться либо с помощью мыши, устанавливая фокус (курсор) щелчком мыши по нужному элементу, либо нажатием клавиши *Tab* на клавиатуре (фокус устанавливается на следующий элемент).

Данные либо вводятся с клавиатуры, либо выбираются из предлагаемых раскрывающихся списков. При вводе значений с клавиатуры контролируется их корректность. Например, контролируется необходимость ввода обязательных значений: при попытке пропустить ввод обязательного для ввода реквизита документа или показателя и при перемещении фокуса с этого элемента при оставшемся незаполненным поле на экран выводится соответствующее сообщение. Пользователь может вернуться к вводу пропущенного значения (щелчком по кнопке «ОК») или отказаться от его ввода («Отмена»), рис. А3.26.

Контроль позволяет предотвратить ошибки при обработке и анализе данных при вычислении результатных показателей в первичных отчетах и формировании сводных отчетов.

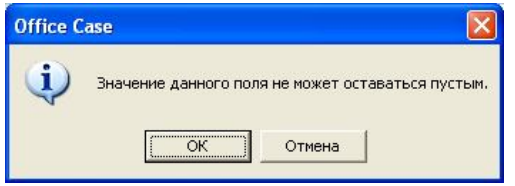


***Рис. А3.26*. Сообщение о некорректно введенном значении**

Если при вводе данных допущенные ошибки не были исправлены (например, вследствие того что необходимые данные оказались неизвестными на момент создания раздела отчета), то всегда можно вернуться к исправлению ошибок, щелкнув по кнопке «Исправить…». Данная команда переводит фокус на первый элемент, при вводе которого была допущена ошибка, и отображает сообщение об этой ошибке, давая пользователю возможность исправить ее. После исправления ошибки можно перейти к следующей и т.д.

Введенные в диалоговых окнах данные можно сохранить в отчете (кнопка «Сохранить»). Сохраненные данные будут доступны для загрузки при последующих обращениях к выбранному разделу отчета. При сохранении введенных данных на экране отображается процесс сохранения (рис. А3.27).

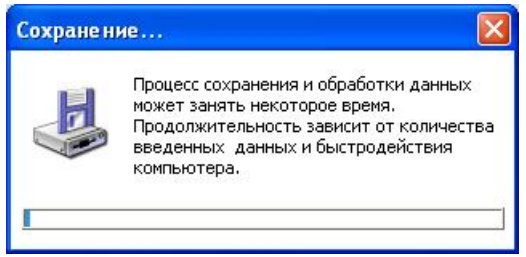


***Рис. А3.27*. Окно, отображающее процесс сохранения данных**

В ходе подготовки к сохранению проверяется правильность введенных данных. Если обнаруживаются неисправленные ошибки, на экран выводится сообщение.

Кроме стандартных элементов Windows в программе используются специальные элементы, позволяющие динамически

изменять число строк в таблицах (это необходимо при включении в таблицы сведений о новых объектах (учредителях, балансодержателях, корпусах и помещениях, воспитанниках и т.п.), информация о которых должна размещаться в отдельных строках таблиц отчета, а их число заранее для каждого конкретного учреждения определить невозможно).

Общий вид этих элементов показан на вкладке диалогового окна, представляющей сведения об учредителях и балансодержателях учреждения оздоровления.

Для добавления новой строки в таблицу отчета необходимо щелкнуть мышью по специальному элементу управления (рис. А3.28), расположенному слева от первой строки. Каждый щелчок по этому элементу добавляет в таблицу отчета по одной строке. Ввести информацию в добавленную пустую строку можно с помощью элементов управления, размещенных в соответствующей строке экранной формы.

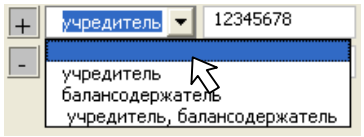


Рис. А3.28. **Дополнительные элементы управления**

Для удалений «лишних» строк из таблицы используются элементы, размещенные слева от каждой вставляемой в таблицу строки. Щелчок по этому элементу полностью удаляет соответствующую строку со всем ее содержимым.

Для очистки первой строки следует выбрать пустое значение в первом элементе строки.

Некоторые элементы отчета связаны друг с другом. Например, код ОКОГУ и наименование министерства (ведомства) или органа управления при определении ведомственной подчиненности учреждения (рис. А3.29).

Для снижения трудоемкости выбор элемента из списка приводит к тому, что автоматически формируется значение связанного поля в отчете, как это показано на рис. А3.29.

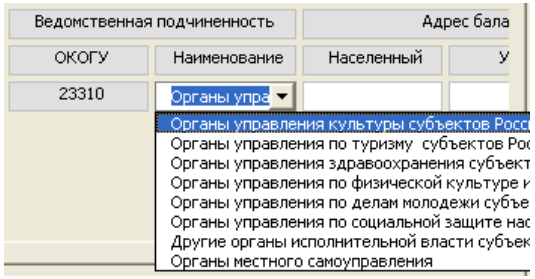


Рис. А3.29. **Связанные элементы управления**

Операции над формируемым разделом отчета в целом выполняются с помощью командных кнопок, расположенных в нижней части диалогового окна редактирования раздела («Сохранить», «Исправить…», «Печатать», «Выход»).

Сформированный отчет можно распечатать на бланке, воспользовавшись кнопкой «Печать». Щелчок по этой кнопке раскрывает диалоговое окно выбора варианта печати. Выбрав нужный вариант, следует щелкнуть по кнопке «ОК». На экране появится стандартное окно печати документа Word.

Щелчок по кнопке «Выход» завершает работу с данным разделом отчета и возвращает пользователя в основной документ с перечнем разделов отчета.

Интерфейс сводных документов

Генератор отчетов для координационных советов построен по принципу «Мастеров»: он позволяет выполнить все шаги построения сводных отчетов, практически не требуя ручного ввода данных: все необходимые сведения выбираются в диалоговых окнах с помощью стандартных элементов управления.

После открытия программы двойным щелчком мыши по значку главного документа отчета «Форма 3.doc» на экране появляется окно.

Работа начинается с создания общего списка учреждений на основе данных, предоставленных учреждениями оздоровления. Для создания списка необходимо щелкнуть по соответствующей кнопке в окне.

После этого открывается первое окно Мастера (рис. А3.30).

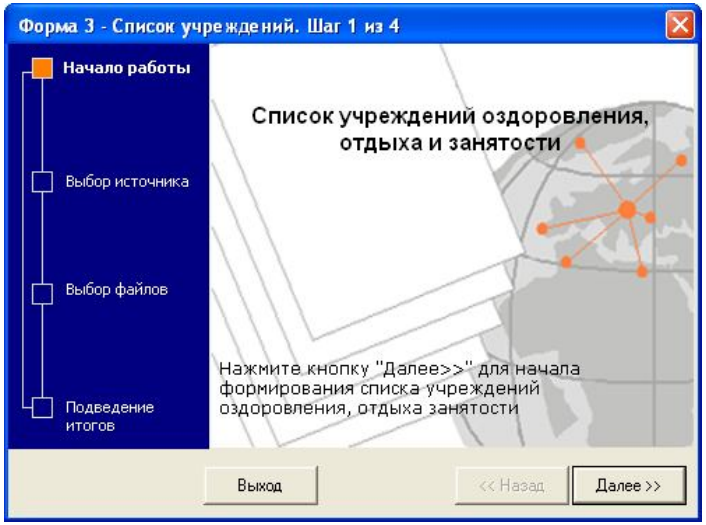


Рис. А3.30. **Мастер, консолидирующий данные первичных отчетов**

Последовательность шагов, которая должна быть выполнена, показывается в левой части окна. Для перехода к следующему шагу нужно щелкнуть кнопку «Далее». Для отказа от выполнения – кнопку «Выход». Диалоговое окно следующего шага («Выбор источника») показано на рис. А3.31.

Выбор источника – это выбор файла (первичного отчета учреждения), из которого нужно взять информацию для включения в отчет. Для поиска папки, в которой размещены первичные отчеты, используется кнопка «Обзор».

В следующем окне можно указать конкретные файлы, информация из которых должна быть использована при создании сводного отчета.

Для выбора файлов с нужными разделами первичных отчетов следует пометить их флажком, установив его щелчком в квадратике, расположенном слева от имени файла (рис. А3.32). Если выбранная папка содержит только нужные разделы отчетов, можно выделить сразу все ее содержимое щелчком по кнопке «Выделить все». Информация из указанных файлов будет включена в сводный отчет. Отменить неверно сделанное выделение можно щелчком по установленному флажку (отменяется только выделение одного файла) или можно снять выделение со всех файлов сразу щелчком по соответствующей кнопке.

Так как «сборка» сводного отчета требует времени, ее ход показывается с помощью специального окна (А3.33).

Если указанные файлы содержат нужную информацию она автоматически агрегируется в сводном отчете. В случае ошибки в первичных отчетах или неверном выборе файлов появится сообщение об ошибке.

Если файлы были выбраны неверно (в приведенном а рисунке примере выделены два разных раздела отчета, содержащие разную по структуре информацию – нужно было бы выделить файлы, представляющие только Раздел I Формы 1), Мастер отчета выводит сообщение об ошибке.

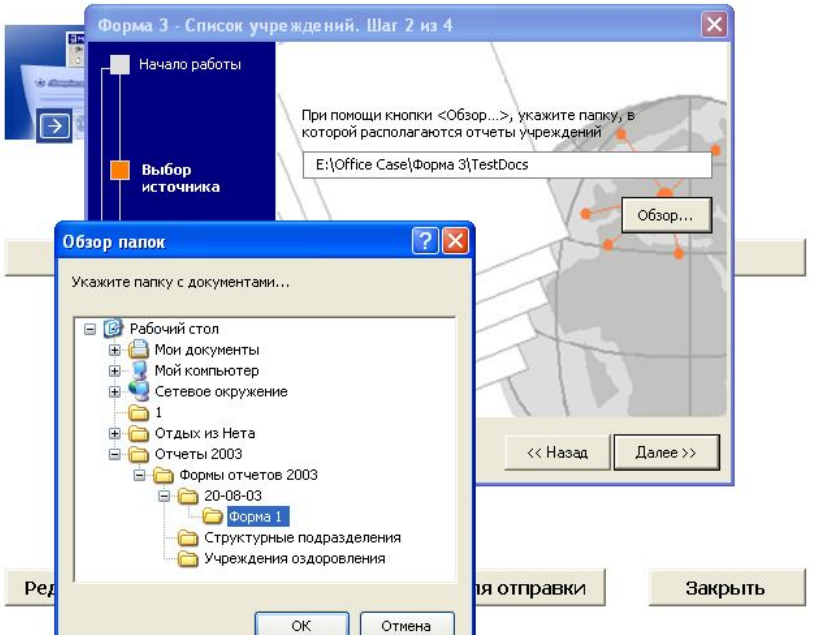


Рис. А3.31. **Выбор источника данных**

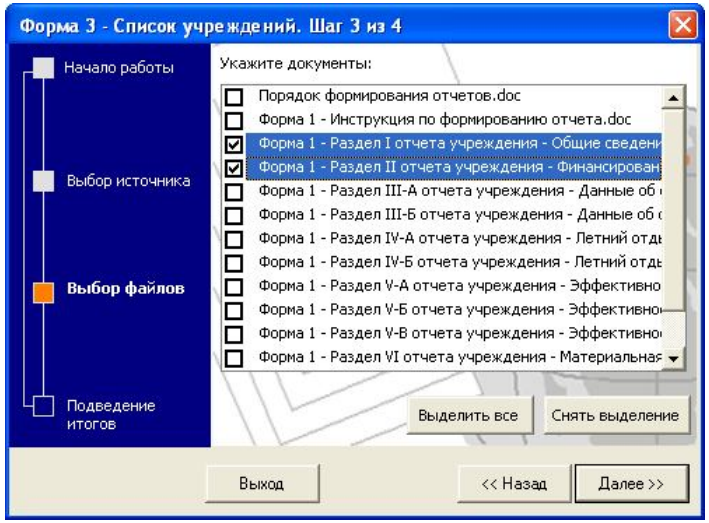


Рис. А3.32. **Выбор первичных документов, включаемых в отчет**

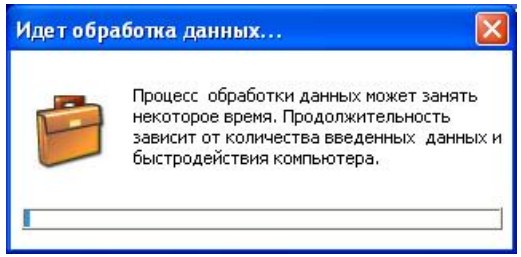


Рис. А3.33. **Окно, отображающее процесс обработки данных**

Сформированные отчеты размещаются в отведенных для этого папках и могут быть переданы по почте, переписаны на дискету или выведены на печать.

Пакет Office CASE – пример созданного на основе Microsoft Office приложения, значительно расширяющего возможности интегрированного офисного пакета по созданию систем взаимосвязанных автоматизированных документов. Описанные выше приемы можно использовать для создания и более простых офисных приложений.

Список литературы

1. *Биллиг В.А.* VBA в Office 2000. Офисное программирование. М.: Издательско-торговый дом «Русская редакция», 1999. – 480 с.: ил.
2. *Вемпен Ф.* Microsoft Office 97 Professional. 6 книг в одной / Пер. с англ. М.: БИНОМ, 1997. 720 с.
3. *Власенко С., Маленкова А.* Word 97 в вопросах и ответах. СПб: BHV–Санкт-Петербург, 1997. 336 с.
4. *Джонс Э., Саттон Д.* Библия пользователя Office 97 / Пер. с англ. Киев: Диалектика, 1997. – 848 с.
5. *Замятина Е.Б., Лядова Л.Н.* Офисные технологии и основы Visual Basic for Application. Пермь: Перм. ун-т, 2001. – 232 с.: ил.
6. *Карлберг К.* Бизнес-анализ с помощью Excel / Пер. с англ. Киев.: Диалектика, 1997. 448 с.
7. *Ланин В.В.* Автоматизация подготовки отчетов на основе приложений Microsoft Office// Математика программных систем: Межвуз. сб. науч. тр. / Перм. ун-т. Пермь, 2002. С. 72-79.
8. *Ланин В.В.* Система автоматизации подготовки документов на основе пакета Microsoft Office // Конференция-конкурс работ студентов, аспирантов и молодых ученых «Технологии Microsoft в информатике и программировании»: Сб. тр. конф. Новосибирск, 2004. С.21-23.
9. *Ланин В.В.* Система управления документами в распределенных средах, основанная на метаданных // Труды 5-ой Всероссийской научно-практической конференции молодых ученых, аспирантов и студентов "Молодежь. Образование. Экономика". Ярославль, 4 мая, 2004.
10. *Ланин В.В., Лядова Л.Н.* Система автоматизации подготовки и обработки отчетов на основе документов Microsoft Office в многоуровневой распределенной системе управления // Международный конгресс конференций «Информационные технологии в образовании». XIII Международная конференция «Информационные технологии в образовании»: Сборник

трудов участников конференции. Часть V. М.: Просвещение, 2003. С. 216-217.

11. *Линекер Р.С., Арчер Т.* Программирование для Windows 98. Библия разработчика / Пер. с англ. М.: Диалектика, 1999. 864 с.
12. *Лоу Д.* Секреты Word для Windows 95. Киев: Диалектика, 1996. 576 с.
13. *Лядова Л.Н., Мызникова Б.И., Фролова Н.В.* Текстовый процессор Word: учебно-методическое пособие. Пермь: Перм. ун-т, 2001. – 172 с.: ил.
14. *Минаси М., Кристиансен Э., Шепер К.* Windows 98: полное руководство / Пер. с англ. Киев: Издательская группа BHV, 1999. 800 с.
15. *Персон Р.* Microsoft Excel 97 в подлиннике: В 2 т./ Пер. с англ. СПб.: BHV–Санкт-Петербург, 1997. Том I. 672 с.
16. *Персон Р.* Microsoft Excel 97 в подлиннике: В 2 т./ Пер. с англ. СПб.: BHV–Санкт-Петербург, 1997. Том II. 640 с.
17. *Санна П.* Visual Basic для приложений (версия 5) в подлиннике. СПб.: BHV–Санкт-Петербург, 1997.
18. *Соломон К.* Microsoft Office 97: разработка приложений. СПб.: BHV–Санкт-Петербург, 1998.
19. *Соломон К.* Microsoft Office: Разработка приложений. СПб.: BHV-Санкт-Петербург, 1998. – 560 с.: ил.
20. *Уокенбах Дж.* Excel-97. Библия пользователя. Киев: Диалектика, 1997. 620 с.
21. *Уэллс Э., Хешбаргер С.* Microsoft Excel 97: разработка приложений. СПб.: BHV–Санкт-Петербург, 1998.
22. *Уэллс Э., Хешбаргер С.* Microsoft Excel: Разработка приложений. СПб.: BHV-Санкт-Петербург, 1998. – 624 с.: ил.
23. *Хэлворсон М., Янг М.* Эффективная работа с Microsoft Office 95 / Пер. с англ. СПб: Питер, 1996. 1024 с.
24. Microsoft Excel 2000: Справочник. СПб.: Изд-во «Питер», 1999.
25. Microsoft Windows 95. Шаг за шагом: Практ. пособие / Пер. с англ. М.: ЭКОМ, 1996. – 320 с.
26. Microsoft Word 2000: Справочник. СПб.: Изд-во «Питер», 1999.

У ч е б н о е и з д а н и е

Лядова Людмила Николаевна
Ланин Вячеслав Владимирович

MICROSOFT OFFICE: ОТ НАЧИНАЮЩЕГО ПОЛЬЗОВАТЕЛЯ ДО ПРОФЕССИОНАЛА

Учебно-методическое пособие

В 2 частях

Часть 2. Основы офисного программирования

Редактор *Г.В. Тулякова*

Корректор *И.А. Михина*

Подписано в печать 05.04.2007. Формат 60×84/16.
Усл. печ. л. 22,5. Уч.-изд. л. 20,5. Тираж 100 экз.
Заказ

Редакционно-издательский отдел Пермского государственного университета
614990. Пермь, ул. Букирева, 15

Типография Пермского государственного университета
614990. Пермь, ул. Букирева, 15