Александр Чиртик
Алексей Гладкий

Excel

Трюки и эффекты

Откройте для себя новые возможности Excel

Алексей Анатольевич Гладкий
Александр Анатольевич Чиртик

Excel. Трюки и эффекты

Аннотация

Данная книга предназначена для пользователей Microsoft Excel и содержит описание приемов и методов работы, которые из-за своей специфичности недостаточно представлены (либо вообще не представлены) в пользовательской, справочной и иной соответствующей документации. Изучение приведенных в книге примеров позволит читателю открыть для себя не известные ранее возможности Excel. Предлагаемый материал легко усваивается благодаря тому, что излагается доступным и понятным языком.

Содержание

Алексей Анатольевич Гладкий, Александр Анатольевич Чиртик Excel. Трюки и эффекты

Введение

В настоящее время табличный редактор Excel, который является разработкой корпорации Microsoft и входит в состав пакета Microsoft Office, – один из самых популярных программных продуктов. Во многом это обусловлено возможностью применения Excel в самых разных отраслях: данную программу используют математики, IT-разработчики, инженеры, экономисты, бухгалтеры, аналитики, менеджеры и т. д. Такое распространение Excel объясняется широкими функциональными возможностями программы и вместе с этим простотой в использовании (удобный и понятный пользовательский интерфейс, возможность быстрого ввода и обработки данных, наглядность представления информации и др.).

Порядок использования программы подробно описывается в справочной подсистеме (для вызова справки достаточ-

но нажать клавишу F1 на клавиатуре или кнопку Справка: Microsoft Offic Excel на ленте в окне приложения). Однако в процессе эксплуатации программы можно также использовать приемы и методы, которые в стандартной документации не рассматриваются либо рассматриваются поверхностно – как правило, потому, что они становятся известны только в результате активной эксплуатации Excel (то есть открываются опытным путем). Иногда они являются сюрпризом даже для самих разработчиков и открывают новые, порой самые неожиданные возможности программы. Описанию подобных трюков и посвящена эта книга.

Большинство описываемых в книге трюков и эффектов выполняется средствами языка VBA (для перехода к редактору VBA используется комбинация клавиш Alt+F11). Книга также содержит описание приемов, выполняемых «подручными» средствами, без программирования.

Структура книги

В главе 1 рассказывается об основах программирования на встроенном в пакет Microsoft Office языке Visual Basic for Applications (VBA). Главный упор сделан на описание синтаксиса и особенностей использования основных конструкций VBA.

В главе 2 рассматриваются нестандартные, но вместе с этим полезные и практичные решения, которые можно реа-

лизовать в рабочей области программы; здесь же рассказывается о приемах работы с файлами Excel (то есть с рабочей книгой). Приводится описание нестандартного использования формул Excel, а также порядок формирования и применения большого количества пользовательских функций. В отдельный раздел вынесено описание трюков и эффектов, которые применяются к ячейкам и диапазонам рабочего листа.

Глава 3 – самая объемная. В ней описывается большое количество самых разнообразных трюков и эффектов, относящихся к различным сферам использования программы. В частности, здесь рассказывается о создании пользовательских меню (как обычных, так и контекстных) и панелей инструментов, о нестандартных приемах форматирования, о работе с примечаниями и др. Большинство приведенных в данной главе трюков имеют практический характер, но есть и такие, реализация которых служит лишь развлекательным либо эстетическим целям.

У пользователей, часто работающих с диаграммами, наверняка вызовет интерес глава 4. В ней рассказывается, как быстро создать диаграмму с помощью макроса, построить диаграмму на основании данных нескольких рабочих листов, быстро изменить тип диаграммы, используя специально созданное контекстное меню, и др.

В главе 5 приведено несколько примеров создания небольших программ – как развлекательных, так и применя-

емых в работе, а также показаны приемы использования созданных программ.

В главе 6 содержится перечень полезных советов, которые пригодятся и начинающим, и опытным пользователям программы. Для удобства восприятия материал представлен в режиме «вопрос – ответ».

В приложении описаны наиболее часто используемые в приведенных в книге примерах стандартные объекты Excel: Application, Chart, Range, Workbook и Worksheet.

Общие положения

При написании книги использовалась версия Excel 2007. Тем не менее большинство приведенных в книге трюков работает и в других версиях программы.

Перед тем как вплотную приступать к изучению нестандартных приемов работы с Excel, кратко вспомним, для решения каких задач предназначено данное приложение и каковы его функциональные возможности, а также определимся с основными терминами и понятиями, которыми мы будем оперировать в дальнейшем.

Подразумевается, что данную книгу будут изучать пользователи, имеющие как минимум начальное представление об Excel. Тем не менее не будет лишним вспомнить, какие задачи решаются с помощью этой программы, а также ознакомиться с используемой в издании терминологией.

Назначение и функциональные возможности Microsoft Excel

Табличный редактор Microsoft Excel предназначен для решения следующих задач.

• Ввод и обработка табличных данных с использованием встроенных механизмов формул, функций, макросов и др.

• Анализ и управление данными (автоматический расчет итоговых и промежуточных данных, структуризация и консолидация данных, использование сводных таблиц, отчетов и др.).

• Импорт необходимых данных из различных источников (включая базы данных OLAP) и последующая их обработка. Поддержка XML-формата.

• Работа с графическими объектами и диаграммами.

• Взаимодействие и обмен данными с программой Lotus Notes, а также интеграция с другими программными продуктами («Галактика», «1С» и др.).

• Работа в Интернете (изменение данных на веб-странице, размещение данных Microsoft Excel в Сети, поддержка веб-файлов, гиперссылок и др.).

• Доступ к данным совместно с другими программами (Word, PowerPoint, Access и др.).

• Формирование самых разнообразных отчетов: аналитических, сводных, графических, в виде диаграмм и др.

• Выполнение стандартных функций Microsoft Office: печать документа, поиск данных и их замена, проверка наличия ошибок, защита информации и др.

• Создание приложений с применением языка программирования VBA.

С помощью Excel можно решать и другие задачи, возникновение которых обусловлено потребностями конкретного пользователя.

Используемая терминология

В данной книге используются следующие основные термины и понятия.

• Автофигура – готовая к использованию фигура заданной формы, которую можно добавлять на рабочий лист или в диаграмму. В Excel имеется встроенный набор автофигур.

• Диаграмма – визуальный способ представления числовых значений. Программа Excel поддерживает работу с различными видами диаграмм: круговыми, пузырьковыми, гистограммами, графиками и др.

• Имя – идентификатор, который предоставляет возможность ссылаться на какой-либо объект (ячейку, диапазон, формулу и т. д.).

• Комментарий – текст, который следует в программном коде сразу после символа «» вплоть до окончания данной строки и игнорируется при выполнении программы. Ком-

ментарий обычно включает в себя произвольную информацию вспомогательного характера, предназначенную для описания и пояснения определенных фрагментов кода либо всего кода.

• Контекстное меню – меню, содержащее список команд, которые предназначены для работы с конкретным объектом. Для вызова контекстного меню нужно щелкнуть на объекте правой кнопкой мыши или нажать комбинацию клавиш Shift +F10.

• Макрос – программа, которая написана на встроенном в Excel языке программирования Visual Basic for Applications (VBA). Переход в режим работы с макросами осуществляется с помощью команды Вид → Макросы.

• Массив – определенное количество ячеек либо значений, с которыми работают как с единым целым. Иначе говоря, массив – это группа элементов одного типа, которые имеют общее имя.

• Модуль – совокупность описаний, инструкций и процедур, сохраненная под общим именем в редакторе VBA.

• Надстройка – программа, внедренная в Excel для расширения функциональных возможностей. Чтобы подключить надстройки, следует в режиме настройки программы в разделе Настройки в поле Управление выбрать значение Настройки Excel и нажать кнопку Перейти, после чего в открывшемся окне установить требуемые флажки.

• Настройка – изменение ныне действующих параметров

работы Microsoft Excel стандартными средствами, доступ к которым осуществляется из рабочего интерфейса Excel. Параметры работы программы можно разделить на два основных вида.

– Общие параметры – редактирование этих параметров приведет к соответствующим изменениям во всех рабочих книгах, в том числе и во вновь создаваемых.

– Локальные параметры – редактирование этих параметров вызовет соответствующие изменения только в текущей книге.

Примечание

Некоторые параметры работы Microsoft Excel можно изменить без использования стандартных средств.

• Область задач – элемент программы, предназначенный для быстрого выбора одной из нескольких связанных задач.

• Панель инструментов – панель, включающая в себя кнопки и иные элементы управления, которые используются для выполнения различных команд. Создание панелей инструментов осуществляется на вкладке Надстройки.

• Печать – вывод содержимого рабочей книги (полностью либо частично) на бумажный носитель с помощью принтера. На печать можно выводить следующие объекты: рабочую книгу, несколько рабочих книг, рабочий лист, несколько рабочих листов, диапазон ячеек на рабочем листе, диапазон ячеек на нескольких рабочих листах, графические объекты, диаграммы. При этом существует возможность вывода на пе-

чать нескольких копий объекта за один сеанс.

• Пользовательский интерфейс – средство взаимодействия пользователя с программой. В состав пользовательского интерфейса входят лента с вкладками, группы, диалоговые окна и др. В Excel применяется стандартный пользовательский интерфейс Windows.

• Примечание – вспомогательная информация произвольного характера, относящаяся к определенной ячейке и хранящаяся независимо от содержимого этой ячейки. Чтобы добавить примечание к какой-либо ячейке, нужно выделить ее и выбрать в контекстном меню пункт Вставить примечание, после чего с клавиатуры ввести требуемый текст.

• Рабочая книга – файл, который создается, редактируется и сохраняется средствами Microsoft Excel. В большинстве случаев рабочая книга имеет расширение XLSX. Основной структурной единицей рабочей книги является рабочий лист (см. ниже).

• Рабочий лист – основной элемент рабочей книги, предназначенный для ввода, редактирования и хранения данных, а также для выполнения вычислений. По умолчанию в состав рабочей книги включены три рабочих листа. Основной структурной единицей рабочего листа является ячейка (см. ниже).

• Редактор VBA – интегрированная среда разработки, в которой осуществляется написание кодов (программирование) на языке VBA. Чтобы перейти в данный режим, необ-

ходимо нажать сочетание клавиш Alt+F11.

• Строка заголовка – стандартный элемент интерфейса многих приложений, расположенный в его верхней части. В данной строке отображается имя открытого документа.

• Строка формул – предназначена для ввода формул и редактирования содержимого ячеек.

• Форматирование – изменение отображения ячейки (ее «внешнего вида») либо представления данных, содержащихся в ячейке. Параметры форматирования ячейки не зависят от ее содержимого, и наоборот. Не стоит забывать, что после применения форматирования отображенное в ячейке значение может не совпадать с ее фактическим значением (наиболее характерный пример – округление: в ячейке хранится значение 0,24, но в соответствии с параметрами форматирования на экране может отображаться значение 0,2).

> **Совет**
>
> Точное фактическое значение, хранящееся в ячейке, при необходимости можно увидеть в строке формул, где оно отображается независимо от параметров форматирования.

• Формула – специальный инструмент Excel, предназначенный для расчетов, вычислений и анализа данных. Формула может включать в себя константу, оператор, ссылку, имя ячейки (диапазона) и функцию (см. ниже).

Операторы бывают трех видов.

– Арифметический оператор – предназначен для выпол-

нения арифметических действий и выдающий в качестве результата числовое значение.

– Оператор сравнения – используется для сравнения данных и выдает в качестве результата логическое значение ИСТИНА или ЛОЖЬ.

– Текстовый оператор – применяется для объединения данных.

• Функция – готовая формула Microsoft Excel для расчетов, вычислений и анализа данных. Каждая функция может включать в себя константу, оператор, ссылку, имя ячейки (диапазона) и формулу. Пользовательская функция – это функция, написанная пользователем на языке VBA.

• Электронная таблица – интерактивная программа, состоящая из набора строк и столбцов, которые выводятся на экран в отдельном окне.

• Ячейка – наименьшая (элементарная) часть электронной таблицы, предназначенная для ввода и хранения информации. Каждая ячейка может содержать текст, число или формулу. Кроме того, при работе с ячейками используются следующие элементы.

– Адрес – это месторасположение (координаты) ячейки. Адрес состоит из буквы (номера) столбца и номера строки, на пересечении которых расположена данная ячейка.

– Ссылка – указание на адрес ячейки. Ссылки могут быть абсолютными (то есть не изменяющимися при перемещении и копировании ячейки), относительными (эти ссылки изме-

няются при перемещении и копировании ячейки) и смешанными. Внешняя ссылка – это ссылка на ячейку, расположенную в другой рабочей книге.

После того как мы вспомнили основные термины и понятия, используемые в Excel, можно приступать к изучению нестандартных приемов и методов работы с данной программой. И в первую очередь мы рассмотрим трюки, выполняемые в рабочей области.

Тексты программ на сайте издательства

Все приведенные в книге листинги (коды программ) можно загрузить с сайта издательства «Питер» по адресу http://www.piter.com/downLoad/978591180547/.

От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты dgurski@minsk.piter.com (издательство «Питер», компьютерная редакция). Мы будем рады узнать ваше мнение!

На сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

Глава 1
Краткое руководство по VBA

Цель данной главы – ознакомить читателя с основами программирования на языке Visual Basic for Applications (VBA), который встроен в пакет Microsoft Office.

Эта глава по форме изложения скорее похожа на справочник. Она рассчитана на пользователей, имеющих некоторый опыт создания программ на других объектно-ориентированных языках программирования, то есть представляющих, что такое переменная, константа, оператор, цикл, массив, класс, объект и т. д. Главный упор в данной главе сделан на описание синтаксиса и особенностей использования основных конструкций VBA, которые нужно знать для понимания приведенных в тексте листингов.

Знакомство с VBA

VBA – это язык программирования, поддерживаемый большинством приложений пакета Microsoft Office. Для запуска среды программирования VBA можно использовать сочетание клавиш Alt+F11.

Возможности VBA

Благодаря высокой степени интеграции в приложения язык VBA позволяет программисту легко применять существующие возможности Microsoft Office для решения специфических задач. Не менее легко VBA позволяет добавлять в приложения этого пакета новые возможности, увеличивая функциональность приложений и удобство работы с ними.

Применительно к Excel программирование на VBA позволяет реализовывать следующие возможности (естественно, это далеко не полный список):

- добавление функций для специфических расчетов;
- ускорение ввода данных в таблицу;
- автоматизацию типичных, часто выполняемых пользователем действий;
- создание команд меню, панелей инструментов как на основе уже имеющихся, так и выполняющих совершенно новые действия;

• повышение наглядности данных (например, с помощью различной окраски ячеек);

• автоматизацию обработки больших объемов данных;

• автоматизацию создания разнообразных отчетов, бланков и прочих действий, связанных с выбором заданной информации из большого объема исходных данных.

Реализация всех этих возможностей при использовании VBA очень часто достигается написанием небольшого количества достаточно простого программного кода.

Структура проекта VBA

Внешний вид редактора VBA с открытым проектом представлен на рис. 1.1.

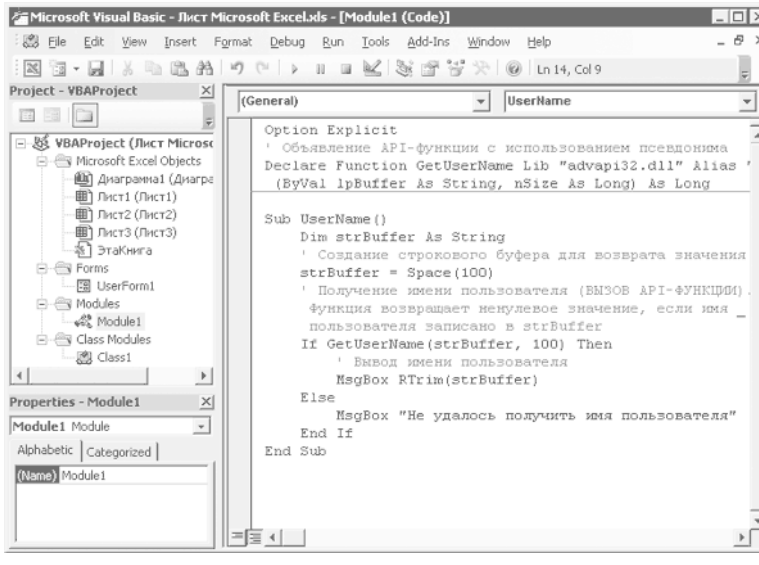

Рис. 1.1. Проект VBA

В редакторе открыты три окна: слева вверху – окно структуры проекта (Project), слева внизу – окно свойств (Properties) и справа – окно с текстом программы модуля. С помощью окна Project (Проект) можно просматривать структуру проекта, добавлять и удалять элементы проекта, открывать для редактирования содержимое модуля (двойным щелчком на значке соответствующего модуля). Окно Properties (Свойства) используется для задания свойств выделенных элементов проекта.

VBA-проект в Microsoft Excel может содержать следую-

щие элементы:

- модули (стандартные модули VBA);
- модули класса;
- модули рабочей книги;
- модули рабочих листов;
- модули диаграмм;
- формы.

Стандартный модуль VBA

Стандартный модуль VBA – это элемент проекта, который содержит программный код, непосредственно используемый остальными элементами проекта (глобальные функции, переменные, константы и т. д.). В окне структуры проекта стандартные модули группируются в папку Modules.

Обычно в стандартном модуле записываются программы, которые не привязаны к конкретным объектам, таким как рабочий лист и рабочая книга. Именно в этом модуле записывается большинство примеров (трюков), рассматриваемых в последующих главах книги.

Модуль класса

Модуль класса – это модуль, в котором записывается программный код, реализующий работу пользовательских (со-

зданных программистом) классов. В окне структуры проекта такие модули группируются в папку Class Modules.

Модуль рабочей книги

Модуль рабочей книги (ЭтаКнига в папке Microsoft Excel Objects) – это модуль класса, в котором реализуются дополнительные возможности по манипулированию рабочей книгой. Программы, записанные в этом модуле, могут напрямую обращаться к свойствам и методам объекта рабочей книги (см. описание модулей класса в конце главы). Отличием модуля рабочей книги от обычного модуля класса является то, что из программы на VBA нельзя создать экземпляр объекта рабочей книги – он создается автоматически.

Модуль рабочего листа

Модуль рабочего листа (папка Microsoft Excel Objects) – это модуль класса, в котором реализуются дополнительные возможности по манипулированию определенными рабочими листами книги. Программы, записанные в этом модуле, могут напрямую обращаться к свойствам и методам объекта рабочего листа. Из программы на VBA также нельзя создать экземпляр объекта конкретного рабочего листа – можно создать только новый рабочий лист с пустым модулем.

Модуль диаграммы

Модуль диаграммы (папка Microsoft Excel Objects) – это модуль класса, в котором реализуются дополнительные возможности по манипулированию диаграммами, вынесенными на отдельные листы рабочей книги. Особенности модуля диаграммы аналогичны особенностям модуля рабочего листа.

Форма

Форма (папка Forms) – это элемент проекта VBA, с помощью которого можно создавать диалоговые окна для взаимодействия с пользователем. Форма состоит из двух частей: модуля формы и собственно формы (диалогового окна).

Модуль формы – это модуль класса, в котором реализуется поведение формы. Использование этого модуля аналогично использованию обычного модуля класса. Для открытия модуля формы служит пункт View Code (Просмотр кода) контекстного меню, которое вызывается щелчком правой кнопки мыши на значке формы в окне Project (Проект).

Для задания внешнего вида диалогового окна используется редактор форм, который открывается с помощью пункта View Object (Просмотр объекта) контекстного меню. Форма,

открытая для редактирования внешнего вида, показана на рис. 1.2.

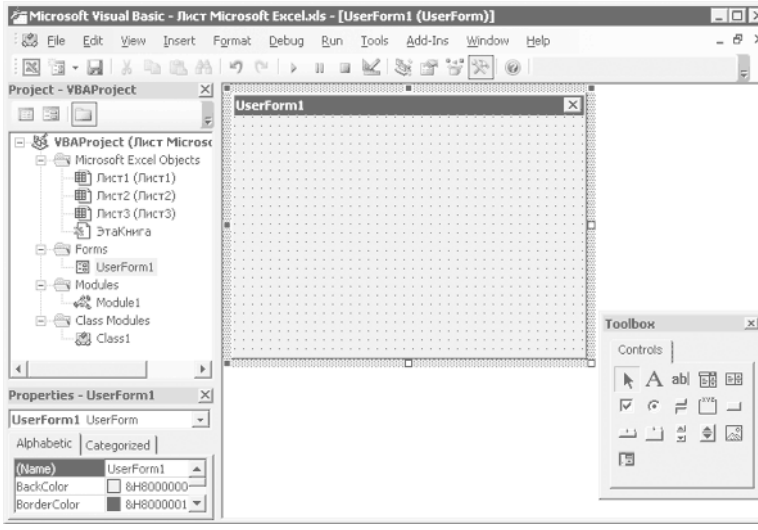

Рис. 1.2. Редактирование формы

Для изменения свойств открытой формы используется окно Properties (Свойства). В окне ТооШох (Панель инструментов) можно выбирать элементы управления, добавляемые в форму. После добавления любого элемента управлениях помощью двойнго щелчка кнопкой мыши на нем можно перейти к редактированию соответствующего кода в модуле формы.

Структура модуля VBA

При разработке любой программы на VBA программный код записывается в одном или нескольких модулях. Код, записанный в любом модуле VBA, имеет следующую структуру.

1. Объявления переменных, директивы (с использованием ключевого слова Option), объявления API-функций.
2. Объявления и реализация процедур и функций.

Соглашения, применяемые при описании синтаксиса VBA

В данном разделе приводится описание элементов, которые используются для формального задания синтаксиса конструкций языка VBA. Сведения об этих элементах приведены в табл. 1.1.

Таблица 1.1. Элементы описания синтаксических конструкций VBA

Элемент	Описание
Значение	Текст, написанный курсивом, обозначает произвольное значение (константное, такое как строка "D:\asd.txt", или значение переменной)
Инструкции	Это слово используется для обозначения произвольной последовательности инструкций программы. Может употребляться с номером, например «Инструкции1». Может также употребляться в составе словосочетания в его начале, например «Инструкции подпрограммы»
Open	Текст, записанный без использования курсива (кроме слова «Инструкции»), является ключевым словом или идентификатором VBA (названием инструкции, оператора, функций и т. д.)
[Необязательный_ Элемент]	Используется для указания необязательных элементов конструкции
Элемент1 | Элемент2	Применяется для указания возможности выбора употребляемого элемента конструкции (то есть использовать или *элемент1*, или *элемент2*)
...	Используется для указания возможности повторения последнего элемента конструкции бесконечное количество раз

Примечание

Кроме описанных в таблице элементов, при задании формата синтаксических конструкций и в текстах программ используется символ подчеркивания «». Он является стандартным символом VBA. Текст, который заканчивается символом подчеркивания, представляет собой одно целое с текстом на следующей строке.

Чтобы сказанное выше стало более понятным, ниже приведен пример формального задания синтаксической конструкции языка VBA. В качестве примера взят формат упрощенного объявления локальной переменной (смысл всех элементов данной конструкции будет полностью раскрыт в последующих разделах главы):

Dim | Static Имя [As Имя_типа][, Имя_переменной [As Имя_типа]]…

Приведенная запись означает, что строка объявления локальной переменной должна начинаться инструкцией Dim или Static. После инструкции должен следовать идеитификатор переменной. Необязательным элементом конструкции является указание типа переменной. Но если тип указывается, то значение в квадратных скобках (первых) должно быть использовано полностью, то есть ключевое слово As не должно применяться без указания имени типа. Объявления переменных можно продолжить в этой же строке без повторного использования инструкции Dim, но разделяя их запятой (см. вторые скобки). Подобные объявления можно продол-

жать в строке до бесконечности (об этом говорит использование многоточия после вторых скобок).

Примеры объявлений переменных, удовлетворяющие указанному формату, приведены ниже:

```
Dim intPos As Integer
Dim varValue, intValue As Integer
Static strText As String
Static var1 As Variant, var2 As Variant, var3 As Variant
```

Комментарии в программе

В VBA предусмотрены два способа введения комментариев в программы. Первый – это использование ключевого слова Rem для обозначения начала комментария. Второй – использование вместо Rem апострофа (). Главным различием этих двух способов является то, что ключевое слово Rem должно находиться в начале строки программы. При этом вся строка является комментарием. Например:

```
Rem Объявление переменной
Dim intRes As Integer
Rem Присвоение значения переменной
intRes = 123
```

Комментарий же, вводимый с помощью апострофа, может быть расположен как в отдельной строке, так и на одной строке с другими инструкциями (в конце этой строки):

```
' Объявление переменной
Dim intRes As Integer
intRes = 123 Присвоение значения переменной
```

Все комментарии в VBA являются однострочными, но при необходимости их текст может быть перенесен на следующую строку с использованием символа подчеркивания:

```
' Длинный комментарий, текст которого не
помещается _
```

в одной строке

или

Rem Длинный комментарий, текст которого не помещается _
в одной строке

Идентификаторы

Идентификаторами в VBA являются названия переменных, констант, функций, процедур, классов, типов данных и прочих элементов, не являющихся зарезервированными словами языка (названиями инструкций, операторов, встроенных функций и т. д.).

Среда разработки VBA поддерживает кодировку символов Unicode. Поддержка данной кодировки разработки означает, что программист может использовать в составе идентификаторов символы любого поддерживаемого алфавита (например, кириллицы).

При формировании идентификаторов необходимо учитывать следующее.

• Идентификатор должен состоять только из букв (любого алфавита), цифр и символа подчеркивания.

• Первым символом идентификатора должна быть буква.

> **Внимание!**
>
> VBA не различает регистр символов в идентификаторах. Это значит, что идентификаторы strmyText и strMyText будут представлять одну и ту же переменную. Это же справедливо и для идентификаторов процедур, функций, классов и т. д.

Рассмотрим примеры корректных идентификаторов VBA:

strText
CUSTOM_Data2
Функция_Суммы
РасчетПрибыли

Переменные

В данном разделе читатель ознакомится с основными особенностями использования переменных при написании программ на языке VBA.

Встроенные типы данных

VBA располагает множеством встроенных типов данных. Условно эти типы можно разделить на численные типы, строки, ссылки, типы для хранения даты и времени, объектные ссылки, массивы и особый тип для хранения значения любого типа, именуемый Variant.

Численные типы данных

Основные характеристики численных типов VBA приведены в табл. 1.2.

Таблица 1.2. Численные типы данных VBA

Тип	Объем памяти (байт)	Диапазон значений
Byte (байт)	1	От 0 до 255
Integer (целое)	2	От –32 768 до 32 767
Long (длинное целое)	4	От –2 147 483 648 до 2 147 483 647
Single (с плавающей точкой)	4	От –3.402 823E38 до –1.401 298E-45, от 1.401 298E-45 до 3.402 823E38
Double (с плавающей точкой двойной точности)	8	От –1.797 693 134 862 32E308 до — 4.940 656 458 412 47E-324, от 4.940 656 458 412 47E-324 до 1.797 693 134 862 32E308
Currency (масштабируемое целое)	8	От –922 337 203 685 477.5808 до 922 337 203 685 477.5807
Decimal (фиксированной длины, повышенной точности)	14	Без дробной части: +/ 79 228 162 514 264 337 593 543 950 335; с максимальной точностью: +/–7.922 816 251 426 433 759 354 395 033 5 (до 28 знаков после запятой)
Variant (для числовых значений)	16	Любое значение из указанных для остальных типов диапазонов

Примечание

Численный тип Decimal как самостоятельный тип на сегодняшний день не поддерживается. Однако его можно использовать в пределах типа Variant (о типе Variant будет рассказано далее).

Строки

Для хранения символьных данных в VBA реализована

поддержка типа данных String (строка). В переменных этого типа могут храниться отдельные символы и большие фрагменты текста. Строки в VB A бывают двух видов: фиксированной и переменной длины. Разница между этими двумя типами строк понятна из их названий.

Строки фиксированной длины применяются, когда длина текста, который хранится в них, постоянна или не может превышать известный предел (например, для хранения отдельных символов). Строки фиксированной длины в ряде случаев обрабатываются быстрее строк переменной длины. Максимальный размер строки фиксированной длины – около 65 400 символов.

Строки переменной длины являются более гибким инструментом обработки текста в программах на VBA. Длина этих строк может динамически изменяться в зависимости от длины хранимого в них текста. Максимальная длина таких строк – около 2 млрд символов.

Дата и время

Для удобства работы со значениями даты и времени в VBA введен тип данных Date. Он позволяет задавать значения времени и даты в формате, удобном для восприятия, а также упрощает вычисления с временными интервалами. Этот тип данных, естественно, используется и в таблицах Excel.

Нужно заметить, что тип Date не является внутренним типом, используемым VBA для хранения даты и времени. Вместо него применяется тип Single (число с плавающей точкой). Целая часть этого числа – количество суток, прошедших с 30 декабря 1899 года, дробная – прошедшая часть текущих суток.

Тип данных Variant

В VBA предусмотрен один универсальный тип данных – Variant. Переменная этого типа может хранить значение любого поддерживаемого VBA типа (в том числе и ссылки на объекты, о которых будет рассказано ниже).

Однако при обработке переменных типа Variant тратится дополнительное время на определение и преобразование типа данных в этих переменных – самый универсальный тип данных VBA оказывается и самым медленным. Поэтому следует избегать слишком частого и неоправданного использования переменных этого типа (например, в качестве целочисленных итераторов, счетчиков и т. д.).

Когда переменная типа Variant пуста (ей не присвоено никакого значения), она заполняется специальным значением Empty.

Ссылки. Тип данных Object

Важно понимать, что в VBA переменные, предоставляющие доступ к объектам, являются только ссылками на эти объекты. В данном языке программирования невозможно получить сам объект (его двоичный код). Все операции по созданию, удалению объектов и манипулированию ими, осуществляются только с использованием ссылок.

Объекты, доступные из VBA, существуют, пока на них установлена хотя бы одна ссылка. Первая ссылка на объект устанавливается при его создании. В процессе работы можно как устанавливать новые ссылки на объект, так и удалять их с использованием специальной инструкции Set. Пусть objRef – ссылка на некоторый объект. Тогда операция установления новой ссылки будет иметь такой вид:

```
Set objNewRef = objRef
```

Теперь objNewRef ссылается на тот же объект, что и objRef. Операция же удаления ссылок будет выглядеть следующим образом:

```
Set objRef = Nothing
Set objNewRef = Nothing
```

Если на объект не было других ссылок, кроме этих, то он будет удален.

Для доступа к объектам в VBA предусмотрен тип дан-

ных Object. Он является универсальным, так как может быть ссылкой на объект любого типа.

Объявление переменных

Для объявления переменных элементарных типов (не массивов) в блоке объявлений модуля используется следующая инструкция:

```
Public | Private [WithEvents] Имя_переменной [As [New] Имя_-
типа] _
[, Имя_переменной [As [New] Имя_типа]]…
```

Ключевые слова, записанные до первых квадратных скобок, задают область видимости переменных:

• Public – позволяет объявлять глобальные переменные и общие переменные-члены класса (о классах будет рассказано позже);

• Private – позволяет объявлять переменные, доступные только в одном модуле, и частные переменные-члены класса.

Для объявления переменных элементарных типов (не массивов) в процедурах или функциях используется такая инструкция (локальных переменных):

```
Dim | Static [WithEvents] Имя_переменной [As [New]
Имя_типа] _
[, Имя_переменной [As [New] Имя_типа]]…
```

Ключевые слова, записанные до первых квадратных ско-

бок, задают время жизни переменных:

• Dim – используется для объявления локальных переменных, которые уничтожаются после выхода из процедуры;

• Static – используется для объявления локальных переменных, значения которых сохраняются между вызовами процедуры или функции.

Ключевое слово WithEvents используется для объявления переменной-обработчика событий объекта.

Имя_переменной – это идентификатор объявляемой переменной.

Имя_типа – название типа данных объявляемой переменной.

Если конструкция [As [New] Имя_типа] не используется, то типом объявляемой переменной автоматически становится тип Variant.

Если используется ключевое слово New, то создается новый объект. New нельзя использовать совместно с WithEvents, а также при объявлении переменной типа Object и если тип Имя_типа не является объектным.

Ниже приведены примеры объявления переменных на VBA:

```
Public intData As Integer
Private intCount As Integer, varData
Dim strText As String
Static a, b, c
Dim objRef As Object
```

```
Dim objCtrl As New Control
```

Внимание!

При объявлении в одной строке нескольких переменных слово As относится только к переменной, непосредственно после идентификатора которой оно следует. Например, при объявлении Dim a, Ь, c As Integer переменные a и b будут иметь тип Variant, а переменная c – тип Integer.

Инициализация переменных

После того как переменная объявлена, VBA производит ее инициализацию указанным ниже образом.

• Переменным численных типов автоматически присваивается нулевое значение.

• Строки переменной длины после объявления являются пустыми (с нулевой длиной). Строки фиксированной длины заполняются нулевыми символами.

• Данные типа Date инициализируются значением 00:00:00 30.12.1899 (это равняется нулю при представлении даты в численном формате, о котором было рассказано выше).

• Все переменные типа Object и подобные (то есть ссылки на объекты определенного типа) принимают значение Nothing, если при их объявлении не создан новый объект (не использовалось New).

Явное и неявное объявление переменных

Рассмотренное выше объявление переменных называется явным.

VBA также поддерживает неявное объявление переменных. Под неявным объявлением подразумевается возможность использования переменной без ее объявления посредством инструкции Dim, Static, Private или Public. При первом обращении к такой переменной для нее автоматически выделяется память и происходит ее инициализация. Следует отметить, что все неявно объявленные переменные имеют тип Variant.

Использование неявного объявления переменных может как упростить написание программ, так и значительно усложнить процесс их отладки.

К примеру, можно очень долго разбираться, почему же после вычисления такого выражения, как dblSalaryAccount = dblSalaryAcount * 10. 5, значение переменной dblSararyAccount стало равным нулю, если до этого оно было равно 5.37 5. Все дело в небольшой ошибке в названии идентификатора переменной, в результате которой создается новая переменная dblSalaryAcount, которая инициализируется нулевым значением. В случае использования явного объявления переменных подобных ошибок в програм-

ме быть не может, потому что каждый раз при обнаружении необъявленного идентификатора VBA выдает ошибку.

Для включения требования обязательного объявления переменных используется директива Option Explicit. Чтобы данная директива добавлялась в каждый новый модуль автоматически, нужно с помощью меню Tools → Options (Сервис → Параметры) редактора VBA открыть диалоговое окно Options (Параметры) и на вкладке Editor (Редактор) установить флажок Require Variable Declaration (Явное описание переменных).

Константы

Как и в любом другом языке программирования, в VBA можно сопоставлять с идентификаторами константные значения. Объявление констант в языке VBA во многом сходно с объявлением переменных. Синтаксис инструкции объявления константы следующий:

[Public | Private] Const Имя_константы [As Имя_типа] = Значение

Два ключевых слова в первых скобках задают область видимости константы:

• Public – используется для объявления глобальных констант;

• Private – используется для объявления констант, доступных только в том модуле, где они объявлены.

По умолчанию, то есть когда не употреблены указанные ключевые слова, константа является Private.

Имя_константы – задает идентификатор константы.

Значение – константное значение (например, «Строка1», 1.245 ит. д.) либо выражение, в число аргументов которого не входят переменные и функции.

Если тип константы не задан, то он автоматически выбирается VBA исходя из ее значения. Ниже приводятся примеры объявления констант:

```
Const PI As Double = 3.14159265359
Public Const MyConstString = «MyConst»
Private Const НазваниеТаблицы As String = «Отчеты»
```

Операторы

Язык VBA содержит большое количество встроенных операторов, которые позволяют выполнять разнообразные действия над всеми встроенными в VBA типами. Операторы и их операнды по определенным правилам составляются в выражения. Данный раздел посвящен описанию операторов, предоставляемых VBA-программисту.

Операторы для работы с численными значениями

Информация обо всех операторах для работы с численными значениями приведена в табл. 1.3.

Таблица 1.3. Операторы для работы с численными значениями

Формат оператора	Описание
Результат = *Выражение*	Оператор присвоения правой части выражения левой
Результат = *Выражение1* + *Выражение2*	Оператор суммы
Результат = *Выражение1* – *Выражение2*	Оператор разности
Результат = *Выражение1* * *Выражение2*	Оператор произведения
Результат = *Выражение1* / *Выражение2*	Оператор деления
Результат = *Выражение1* \ *Выражение2*	Оператор целочисленного деления (возвращает целую часть от результата деления)
Результат = *Выражение1* Mod *Выражение2*	Оператор, возвращающий остаток от деления
Результат = *Выражение1* ^ *Выражение2*	Оператор возведения в степень. Основание степени — *Выражение1*, показатель степени — *Выражение2*

Примечание

Оператор «+» может использоваться и для соединения строк. Однако bVBA существует специальный оператор «&», выполняющий эту функцию. Рекомендуется использовать для соединения строк именно оператор «&», так как это способствует легкому визуальному отделению операций над строками от операций над другими типами данных, что, в свою очередь, улучшает читаемость кода.

Операторы сравнения

Результатом выполнения всех операторов сравнения является значение типа Boolean. Если операнды какого-либо оператора удовлетворяют его условию, то возвращается значение True, иначе возвращается значение False. Все операторы сравнения, поддерживаемые VBA, приведены в табл. 1.4.

Таблица 1.4. Операторы сравнения

Формат оператора	Описание
Результат = Выражение1 = Выражение2	Оператор равенства. Возвращает True, если значения выражений равны
Результат = Выражение1 > Выражение2	Оператор «больше». Возвращает True, если значение *Выражение1* больше значения *Выражение2*
Результат = Выражение1 < Выражение2	Оператор «меньше». Возвращает True, если значение *Выражение1* меньше значения *Выражение2*
Результат = Выражение1 >= Выражение2	Оператор «больше либо равно». Возвращает True, если значение *Выражение1* больше или равно значению *Выражение2*
Результат = Выражение1 <= Выражение2	Оператор «меньше либо равно». Возвращает True, если значение *Выражение1* меньше или равно значению *Выражение2*
Результат = Выражение1 <> Выражение2	Оператор неравенства. Возвращает True, если значения выражений не равны

Описанные операторы сравнения могут принимать в ка-

честве операндов значения выражений любого типа, то есть фактически оперируют с типом данных Variant. Если один из операндов равен Empty, то результатом выполнения операторов будет специальное значение NULL. Если операнды несравнимы, то при выполнении описанных выше операторов генерируется ошибка: «Несоответствие типа».

Режим сравнения строковых значений можно задать с помощью директивы Option Compare. Для Excel работают два варианта данной директивы: Option Compare Text (текст сравнивается без учета регистра символов) и Opt ion Compare Binary (сравниваются бинарные коды символов, при этом автоматически учитывается их регистр). По умолчанию сравнение строк происходит согласно директиве Option Compare Binary.

В VBA реализованы два специфических оператора Like и Is, которые тоже относятся к операторам сравнения. Они также возвращают значение типа Boolean как результат сравнения.

Оператор I s используется для определения, являются ли две ссылки ссылками на один и тот же объект. Этот оператор допускает использование в качестве операндов только ссылки на объекты. Формат данного оператора такой:

```
Результат = Ссылка1 Is Ссылка2
```

Оператор Like используется для проверки, удовлетворяет ли текст в строке заданному шаблону. Формат этого оператора следующий:

Результат = Строка Like Шаблон

Перечень символов, которые могут употребляться в строке шаблона, приведен в табл. 1.5.

Таблица 1.5. Перечень возможных символов в строке шаблона

Символы в шаблоне	Соответствующие символы в строке
*	Любое количество любых символов

Символы в шаблоне	Соответствующие символы в строке
?	Любой одиночный символ
#	Любая одиночная цифра
[*Список_символов*]	Любой одиночный символ из заданного списка
[!*Список_символов*]	Любой одиночный символ, не входящий в заданный список

В качестве примера определим, является ли строка «15 26 ОА» номером автомобиля серии ОА или ОО. Значение строки-шаблона для этого случая будет равно «## ## О[АО]», а результатом применения оператора «15 26 ОА» Like «## ## О[АО]» будет значение True.

Для задания непрерывного диапазона символов в квадратных скобках можно воспользоваться знаком «минус» (-). При этом символы необходимо указывать в возрастающем порядке (по номеру в алфавите): [A-Z], а не [Z-A].

Примечание

Если нужно, чтобы в шаблоне присутствовали

специальные символы, приведенные в табл. 1.5, то необходимо заключить соответствующие знаки в скобки: ([), (]), (#), (?), (*).

Логические операторы

В VBA введены операторы, которые используются в составе логических выражений (например, условие в инструкции If-Then-Else, которая будет рассмотрена позже). Формат логических операторов VBA (кроме импликации) и их описание приведены в табл. 1.6 (все выражения, используемые в операторах, – логические, принимающие значение True или False).

Таблица 1.6. Логические операторы VBA

Формат оператора	Описание
Результат = *Выражение1* Eqv *Выражение2*	Эквивалентность. Возвращает True, если значения выражений равны
Результат = *Выражение1* And *Выражение2*	Логическое И. Возвращает True, если оба выражения имеют значение True
Результат = *Выражение1* Or *Выражение2*	Логическое ИЛИ. Возвращает True, если как минимум одно из выражений имеет значение True
Результат = *Выражение1* Xor *Выражение2*	Исключающее ИЛИ. Возвращает True, если выражения имеют противоположные значения
Результат = Not *Выражение*	Логическое НЕ. Возвращает значение, противоположное значению выражения *Выражение*

Массивы

При создании программ часто приходится оперировать большими количествами данных одного типа и имеющих одинаковый смысл. Для хранения таких данных используются массивы. Массив – это совокупность значений одного типа, объединенных в одной переменной.

Язык VBA предоставляет широкие возможности для использования массивов. В нем работа с массивами значительно упрощена. Например, при выполнении программы автоматически контролируется выход за пределы массива. Также VBA-программисту при работе с массивами не нужно заботиться о выделении и освобождении памяти.

Объявление массива

Для объявления массивов в VBA используются инструкции, формат которых приведен ниже:

```
Public | Private Имя_массива ([Размерность])[As Имя_типа] _
[, Имя_массива ([Размерность]) [As Имя_типа]]…
```

или

```
Dim | Static Имя_массива ([Размерность])[As Имя_типа] _
```

[, Имя_массива ([Размерность]) [As Имя_типа]]…

Первая инструкция используется для объявления массивов на уровне модуля, вторая – для объявления массива в процедуре или функции (все аналогично объявлению переменных).

Объявление массива отличается от объявления любой другой переменной тем, что при объявлении массива после идентификатора переменной указывается размерность. Если размерность не указана, то создается динамический массив, размер которого можно изменять во время выполнения программы. Для динамического массива инструкция объявления является формальной: чтобы этот массив можно было использовать, к нему необходимо применить инструкцию ReDim (об этой инструкции будет рассказано далее).

При указании размерности массива необходимо учитывать, что элемент Размерность имеет следующий формат:

```
    Нижняя_граница  To  Верхняя_граница  |
Количество_элементов _
    [,Нижняя_граница  To  Верхняя_граница  |
Количество_элементов]…
```

В VBA разрешено создавать многомерные массивы с количеством измерений не более 60. Размерности измерений массива разделяются запятой.

При задании размерности в виде Нижняягранца To Верхняягранца нужно явно указывать нижнюю и верхнюю границы измерения массива (например, 50 To 100).

При задании размерности можно также просто указывать требуемое количество элементов в данном измерении массива. При таком задании в качестве нижней границы измерения используется значение по умолчанию (об изменении этого значения будет рассказано далее).

Ниже приведены примеры объявлений массивов (переменного размера, двух одномерных и двух многомерных):

```
Dim avarValues()
Dim astrValues(1 To 10) As String, astrValues2(10) As
String
Dim aintValues(1 To 10, 1 To 3) As Integer,
aintValues(10, 3)
As Integer
```

Задание нижней границы по умолчанию

Как было сказано ранее, при указании размерности измерения массива может использоваться значение нижней границы по умолчанию. Для задания нижней границы, используемой по умолчанию, предназначена директива Option Base. Существуют только два варианта данной директивы:

```
Option Base 0
```

и

```
Option Base 1
```

Первый вариант устанавливает нижнюю границу равной

нулю (используется по умолчанию), а второй – единице.

Изменение размера массива

Язык VBA позволяет изменять размер динамического массива во время выполнения программы. Кроме того, VBA дает возможность изменять количество измерений такого массива. Для этого используется инструкция ReDim, формат которой следующий:

```
ReDim [Preserve] Имя_массива ([Размерность])[As
Имя_типа] _
[, Имя_массива ([Размерность]) [As Имя_типа]]…
```

Назначение элементов данной инструкции полностью аналогично назначению одноименных элементов инструкции Dim (при использовании ее для объявления массивов). Тип элементов массива можно указывать только в том случае, если Имямассива – это идентификатор переменной типа Variant.

При выполнении инструкции ReDim без использования ключевого слова Preserve значения всех элементов, которые ранее были в массиве, теряются. Ниже приведены примеры таких инструкций:

```
ReDim astrValues(1 To 10), aintValues(10, 20)
ReDim varArray(2 To 4) As Boolean
```

Использование Preserve позволяет изменять размер мас-

сива, не теряя значений его элементов. Однако использование данного ключевого слова налагает некоторые ограничения на возможности манипулирования массивами:

• нельзя изменять количество измерений массива;

• нельзя изменять размерности измерений массива, кроме размерности последнего измерения;

• можно изменять только верхнюю границу последнего измерения массива.

Давайте рассмотрим пример использования инструкции ReDim с ключевым словом Preserve:

```
' Первая инструкция ReDim для динамического массива
ReDim astrValues(1 To 5, 1 To 10)
…
' Увеличение размера массива
ReDim Preserve astrValues(1 To 5, 1 To 25)
…
' Уменьшение размера массива
ReDim Preserve astrValues(1 To 5, 1 To 15)
```

Определение границ массива

Так как VBA позволяет задавать произвольную нижнюю границу массива, при написании программ крайне удобно наличие возможности узнать границы массива во время выполнения программы. Для этой цели в VBA введены две

функции, формат которых следующий:

```
LBound(Имя_массива[, Номер_измерения])
RBound(Имя_массива[, Номер_измерения])
```

Функция LBound позволяет получить нижнюю границу массива, а RBound – верхнюю. Обе функции принимают в качестве аргументов идентификатор массива и номер измерения, границу которого нужно получить. Нумерация измерений начинается с единицы. Если параметр Номеризмерения опущен, то его значение принимается равным единице. Обе функции возвращают значение типа Long.

Ниже приведен пример получения нижней и верхней границ первого измерения массива avarValues (значения сохраняются в переменных типа Long):

```
lngLBound = LBound(avarValues)
lngRBound = RBound(avarValues)
```

Доступ к элементам массива

Для доступа к элементам массива в VBA используется указание номера этого элемента в круглых скобках после идентификатора переменной массива. При этом номера измерений массива разделяются запятыми. Например (для одномерного и трехмерного массивов):

```
intNum = aintValues(16)
intNum = aintValues(12, 32, 3)
```

Использование переменной Variant при работе с массивами

Язык VBA поддерживает универсальный тип данных Variant, которому находится применение и при работе с массивами. Переменной этого типа можно присваивать массив. В результате этой операции в переменной Variant формируется копия массива. Далее с такой переменной можно работать либо как с обычной переменной, либо как с массивом (использовать доступ к элементам), например:

```
Dim aintValues(1 To 3) As Integer
Dim varArray
' Присвоение массива переменной типа Variant
varArray = aintValues
' Доступ к элементам массива
varArray(1) = 1
varArray(2) = 2
varArray(3) = 3
```

Возможность присвоения массива переменной типа Variant на самом деле широко используется в VBA при передаче массивов в функции и процедуры, а также при возврате функциями массивов.

Для определения того, содержит ли переменная типа Variant массив, можно использовать функцию IsArray, имеющую следующий формат:

```
IsArray(Переменная)
```

Данная функция возвращает значение типа Boolean: True – если в переменной с именем Переменная содержится массив, и False – в противном случае.

Использование функции Array для заполнения массива

В VBA имеется возможность быстрого заполнения массива значениями. Эта возможность реализована в функции Array. Ее формат такой:

```
Array(Список_элементов)
```

В качестве аргументов функция принимает список значений, разделенных запятой. Возвращает она заполненный заданными значениями массив, сохраненный в переменной типа Variant. Ниже приведен пример использования функции Array:

```
Dim varArray
' Заполнение массива значениями
varArray = Array(1, 2, 3, 4, 5)
```

Коллекции

Коллекции (они же семейства и множества) – это объекты, которые позволяют хранить произвольное количество элементов любого типа. Элементы в коллекции идентифицируются уникальным ключом, которым может быть не только номер элемента в коллекции, но и значение строкового или другого типа. При программировании на VBA различные коллекции используются очень часто. Например, к коллекции Workbooks нужно обращаться для получения ссылки на объект Workbook нужной рабочей книги, к коллекции Worksheets – для получения ссылки на объект Worksheet нужного рабочего листа и т. д.

В VBA коллекции реализованы во встроенном классе Collection. Создание объекта Collection ничем не отличается от создания объекта другого типа:

```
Dim col As New Collection
```

или

```
Dim col As Collection
Set col = New Collection
```

Добавление элементов

Для добавления элементов в коллекции реализован метод

Add, имеющий следующий формат:

Ссылка. Add Элемент [, Ключ][, Добавить_перед][, Добавить_после]

Единственным обязательным параметром метода Add является значение добавляемого элемента. Элемент может быть константой или переменной любого типа, кроме типа, определенного пользователем. При добавлении элемента можно указать ключ, который будет однозначно идентифицировать элемент в коллекции. Ключ – это любое значение типа Variant.

По умолчанию новые элементы добавляются в конец коллекции. Для изменения порядка добавления элементов используются параметры Добавить_перед и Добавить_после, с помощью которых указывается номер или ключ того элемента, перед которым или после которого нужно вставить новый элемент. Нумерация элементов в коллекции начинается с единицы.

Конец ознакомительного фрагмента.