

Министерство образования Российской Федерации
Южно-Уральский государственный университет
Кафедра информатики

681.3.06(07)
Г699

Е.Н. Горных

ПРОГРАММИРОВАНИЕ НА VBA

Учебное пособие

Челябинск
Издательство ЮУрГУ
2003

УДК 681.3.066(075.8) + [681.3.06:519.6](075.8)

Горных Е.Н. Программирование на VBA: Учебное пособие. — Челябинск: Изд. ЮУрГУ, 2003. — 76 с.

Пособие предназначено для студентов, изучающих курсы, связанные с использованием офисных приложений, а также всем, кто желает познакомиться с основами офисного программирования.

Пособие содержит сведения, необходимые для начального знакомства со средствами автоматизации, встроенными в электронные таблицы Excel. Порядок изложения материала, иллюстрации и практические задания позволяют использовать пособие для самостоятельной работы студентами всех форм обучения.

Ил. 31, список лит. — 10 назв.

Одобрено учебно-методической комиссией факультета экономики и управления.

Рецензенты: Извеков Ю.А., Самохвал О.В.

© Издательство ЮУрГУ, 2003.
© Е.Н. Горных, 2003.

ВВЕДЕНИЕ

Стандартные возможности продуктов Microsoft Office очень значительны и интересны. Они раскрываются для пользователя не сразу, а только в процессе постоянной работы и решения все новых и новых задач, в процессе поиска наиболее быстрого и эффективного способа реализации поставленной цели.

Благодаря встроенному языку программирования Visual Basic for Application (VBA) приложения, входящие в состав Ms Office, превратились в среду разработки программных проектов, в которых можно использовать все мощные функции отдельных составляющих пакета, организовав их совместную работу и взаимодействие с пользователем.

Углубленное изучение Ms Excel и знакомство с основами программирования на VBA, несомненно, даст толчок к появлению новых идей о возможных областях применения ресурсов данной среды.

В последние годы появилось достаточно много литературы, посвященной программированию на VBA, рассчитанной на различный уровень подготовки пользователей. Например, книги В.А. Биллига [1, 2] рассчитаны на подготовленного пользователя, книги А. Гарнаева [3, 4] базируются на опыте преподавания Ms Excel студентам экономических специальностей.

Данное учебное пособие также базируется на многолетнем опыте преподавания Ms Excel и программирования на VBA студентам экономических специальностей. Все примеры и задания, приведенные в пособии, многократно выполнялись студентами на занятиях в компьютерных классах в среде Ms Excel, начиная с пятой версии, конечно, с учетом особенностей каждой версии.

Ограниченный объем пособия не позволил более подробно остановиться на вопросах разработки форм рабочего листа. Этим вопросам посвящена книга Уэллса [9]. В пособии содержится минимально необходимый набор сведений, позволяющий выполнить предлагаемые задания. Поэтому пособие будет полезно и студентам заочного отделения для первого краткого знакомства с предметом. Более подробное описание типового лабораторного задания и других примеров программирования на VBA, будет приведено в следующем учебном пособии, запланированном на 2004 год.

Автор будет признателен читателям пособия за замечания и предложения, которые можно направлять по адресу: Helen@inf.susu.ac.ru.

ПРОСТЕЙШАЯ АВТОМАТИЗАЦИЯ

Макросы

Постепенно углубляя свои знания по использованию Microsoft Excel, и применяя электронные таблицы для обработки данных различного типа, Вы непременно заметите, что для получения тех или иных результатов каждый раз выполняется одна и та же последовательность операций. Очевидно, что возникает вопрос: «Можно ли сделать так, чтобы данная последовательность команд выполнялась одним щелчком мыши (одной командой)?».

Раньше такая предварительно записанная последовательность команд называлась *макрокомандой*. В современном компьютерном языке слово сократилось до простого *макрос*, или, по-английски, *macro*.

При записи макроса программа записывает все действия пользователя, включая ошибочные и лишние. Записанный макрос не гибок, поскольку не учитывает изменение обстоятельств и условий. Макрос не универсален, т.к. действия выполняются точно в том порядке, в котором они были записаны. Старые программы для записи макросов имели существенный недостаток: даже небольшое изменение можно было внести, только повторив все действия еще раз. Это значительно снижало эффективность применения макросов.

Учитывая пожелания пользователей, были созданы средства для редактирования записанных макросов. Языки макросов стали включать в себя возможности, которые ранее встречались только в языках программирования. Однако языки макросов для различных приложений значительно отличались друг от друга, что затрудняло работу пользователей.

Для унификации языков макросов различных приложений компания Microsoft создала специальную версию языка Visual Basic, назвав ее Visual Basic for Application (сокращенно VBA). Excel 5 был первым продуктом, включавшим поддержку VBA. Начиная с Office 97 язык VBA был включен во все приложения.

С помощью VBA можно управлять выполнением других программ и организовать обмен данными между приложениями, используя OLE.

Microsoft Excel можно рассматривать как средство разработки информационных систем, совмещающее в себе преимущества электронных таблиц и средства визуального программирования.

Язык VBA в основном совпадает с Visual Basic for Windows, но имеет и существенные отличия. Например, макросы VBA хранятся в файле того приложения, в котором они были созданы, а не в отдельном текстовом файле; запустить макрос можно только из той программы, в которой он был написан, т.к. каждое приложение вносит в свои макросы специфические команды и объекты.

Запись макросов в автоматическом режиме

Как правило, создание макроса разбивается на несколько этапов.

1. Определить последовательность действий, которая должна быть оформлена в виде макроса и создать начальные условия для их выполнения. Возможно, потребуется пробное выполнение команд для того, чтобы убедиться в их корректности и исключить «лишние» действия.

2. Начать запись макроса, выполнив команду **Сервис – Макрос – Начать запись**. Откроется диалоговое окно «*Запись макроса*», в котором необходимо указать параметры макроса.

— Текстовое поле «*Имя макроса*». Начинается с буквы, но может содержать цифры. В имени не допускается использовать пробелы и знаки пунктуации. По умолчанию предлагается стандартное название макроса, но лучше записать более осмысленное, соответствующее действиям макроса.

— Текстовое поле «*Сочетание клавиш*». Можно указать комбинацию клавиш, с помощью которой макрос будет запускаться на выполнение. Это имеет смысл делать для часто используемого макроса. Комбинация клавиш — это «CTRL» и какой-нибудь символ. Например: «CTRL+a».

— Список «*Сохранить в*». В этом списке надо выбрать, где будет храниться новый макрос. Если выбрать «Эта книга», то макрос запишется в текущей рабочей книге. Вариант «Личная книга макросов» позволит записать макрос в специальной книге под названием **Personal.xls**, которая открывается при каждом запуске Excel, т.е. макрос будет доступен разным рабочим книгам. Вы можете и не заметить, что книга открыта, т.к. по умолчанию **Personal.xls** скрыта. Вариант «Новая книга» заставит Excel открыть новую книгу и сохранить в ней макрос, хотя активной останется прежняя книга и все ваши действия будут применены именно к ней.

— Текстовое поле «*Описание*». В этом поле можно кратко описать назначение макроса и начальные условия его выполнения. Не пренебрегайте этим полем, надеясь на свою память, даже при небольшом количестве создаваемых макросов.

После щелчка на ОК Excel откроет панель «*Остановить запись*», а в строке состояния появится сообщение «*Запись*». С этого момента все действия, выполняемые мышью или с помощью команд, будут записаны в виде инструкций VBA.

3. Необходимо аккуратно выполнить все действия, оформляемые в виде макроса.

4. Остановить запись, выполнив команду **Сервис – Макрос – Остановить запись** или щелкнув на кнопке «*Остановить запись*».

На панели инструментов «*Остановить запись*» имеется еще одна кнопка – «*Относительная ссылка*». По умолчанию в макросах записываются абсолютные номера ячеек, т.е. действия будут выполняться в одних и тех же областях рабочего листа.

Если нажать кнопку «*Относительная ссылка*», то действия будут записываться в относительных адресах, т.е. с учетом текущего положения активной ячейки. Это переключатель, имеющий два положения – приподнятое и утопленное. Каждый щелчок на кнопке переводит режим записи в противоположный. Такое переключение можно выполнять в любой момент на протяжении записи макроса.

Созданный макрос будет записан в особой части рабочей книги, называемой *модулем*. Рабочая книга может содержать несколько модулей. В каждом модуле могут быть записаны несколько макросов. Модуль для записи макроса выбирается автоматически, а при необходимости создается новый. В каждом сеансе работы, если выполняется запись макроса, создается новый модуль. Модули рабочей книги имеют имена ModuleN, где N – порядковый номер модуля. Познакомиться с текстом записанного макроса можно в окне редактора VBA (см. далее пункт «Редактирование макросов»).

Способы запуска макросов

Один из способов запуска макросов был уже рассмотрен при описании поля «*Сочетание клавиш*», т.е. это комбинация клавиш, назначенная при создании макроса.

Существует еще несколько вариантов запуска макросов.

1. Самый простой и очевидный способ – выполнить команду **Сервис – Макрос – Макросы**. Откроется окно «**Макрос**», в котором пользователь должен выбрать макрос в списке (вот когда пригодятся осмысленные названия макросов!) и щелкнуть на кнопке «*Выполнить*».

Если вы все же решили назначить макросу комбинацию клавиш или ввести его описание, то это можно сделать, щелкнув на кнопке «*Параметры*».

Кнопка «*Войти*» переведет вас в режим пошагового выполнения макроса.

Кнопка «*Изменить*» откроет окно редактирования текста макроса.

2. Макрос можно назначить кнопке на панели инструментов, т.е. можно создать свою собственную панель инструментов или добавить кнопку к существующей (предпочтительнее первый вариант, чтобы не вносить изменений в вид стандартных панелей).

Для создания личной панели инструментов необходимо выполнить команду Вид – Панели инструментов – Настройка. Откроется окно настройки панелей инструментов (рис.1).

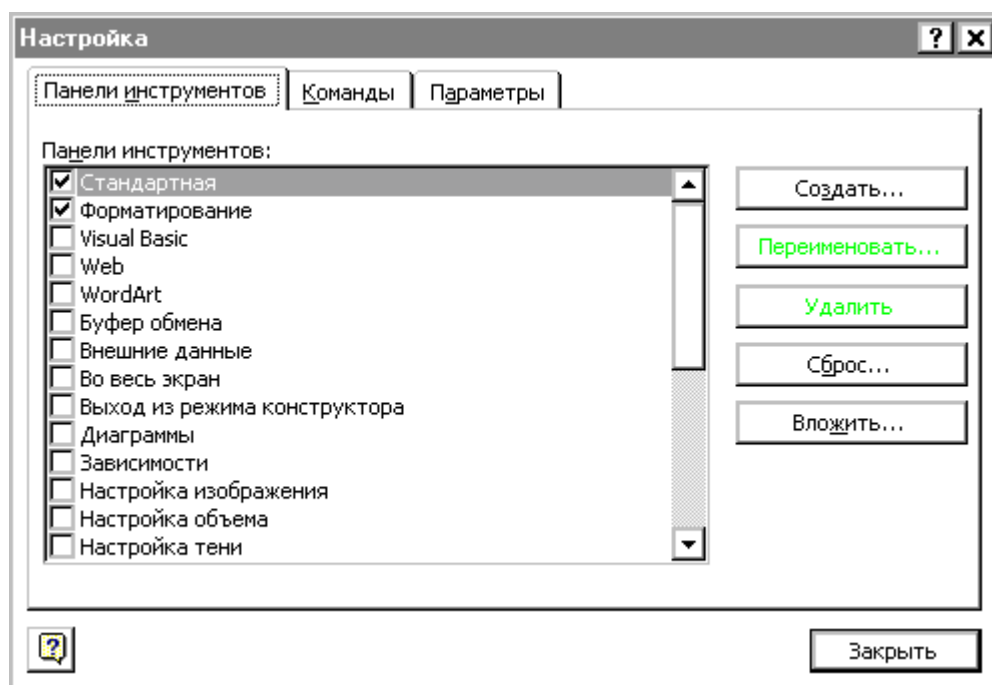


Рис. 1. Настройка панелей инструментов

Затем щелкните кнопку «*Создать*» и введите название личной панели инструментов (оно не должно совпадать с существующими именами). На экране появится небольшое окно новой панели инструментов, не содержащее кнопок. Следующий шаг – добавление кнопок на панель. Переключитесь на вкладку «*Команды*».

На новую панель можно перетащить любую команду из существующих категорий команд.

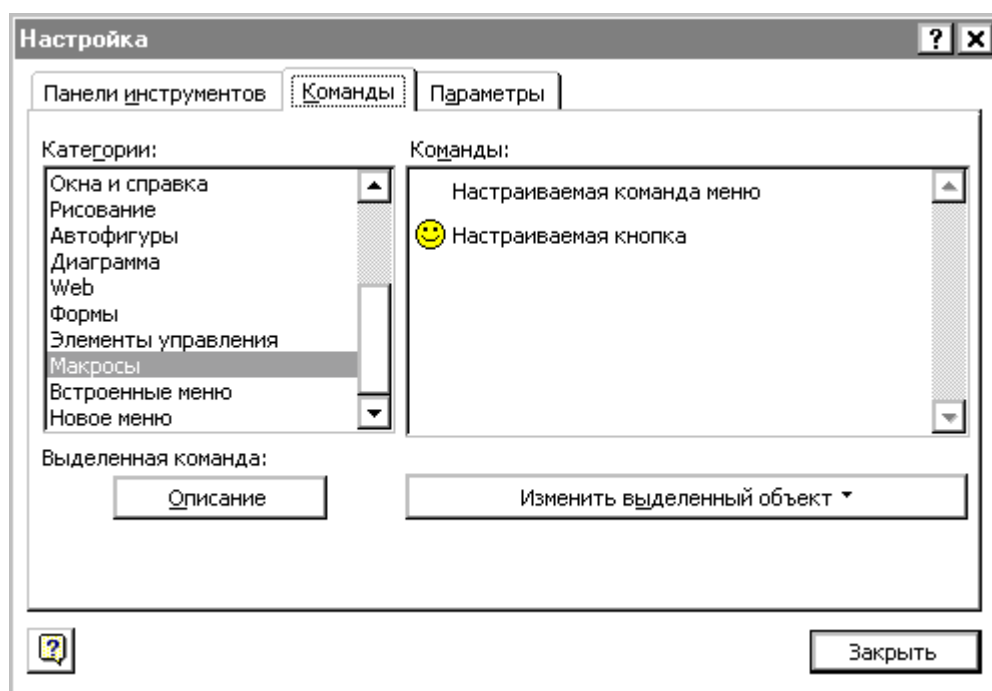


Рис. 2. Добавление кнопок на панель инструментов

Для личного макроса перетащите на панель команду «*Настраиваемая кнопка*» из категории «*Макросы*». Затем щелкните в окне на кнопке «*Изменить выделенный объект*» или откройте контекстное меню кнопки на панели инструментов (правой кнопкой мыши). Введите новое имя кнопки, заменив стандартное. Это имя будет появляться рядом с кнопкой в виде всплывающей подсказки. С помощью команды «*Назначить макрос*» можно подключить к кнопке один из существующих макросов. Если макрос еще не создан, это можно сделать позже. Для того, чтобы кнопки для разных макросов не были одинаковыми, используйте пункты «*Выбрать значок для кнопки*» и «*Изменить значок на кнопке*» для изменения и создания личных кнопок.

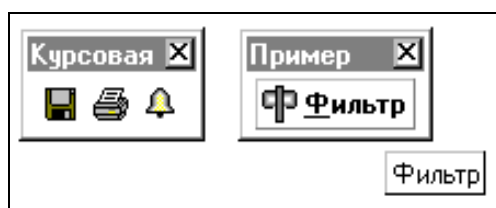


Рис. 3. Примеры личных панелей инструментов

На панели «*Курсовая*» расположены две стандартные кнопки и одна для личного макроса (колокольчик). Используется основной стиль отображения кнопок. На панели «*Пример*» расположена кнопка, созданная в редакторе кнопок, для макроса «*Фильтр*». Для ее отображения установлен стиль «*Значок и текст*». При наведении указателя мыши на кнопку отображается всплывающая подсказка (имя кнопки).

Для подключения макроса к кнопке необходимо выбрать пункт «*Назначить макрос*» в контекстном меню кнопки, выбрать макрос в списке и подтвердить действие. Если для кнопки уже был назначен макрос, то ссылка на него отобразится в поле «*Имя макроса*».

Новая панель инструментов присутствует только на том компьютере, на котором она была создана. Чтобы панель инструментов была связана с рабочей книгой, необходимо:

- открыть окно настройки панелей инструментов (см. рис. 1);
- щелкнуть на кнопке «*Вложить*», откроется окно «*Управление панелями инструментов*»;
- выделить нужные панели в списке панелей пользователя и скопировать их в список панелей книги;
- завершить команду.

После указанных действий созданная панель инструментов будет появляться на любом компьютере при открытии файла.

3. На рабочем листе можно разместить элемент управления, используя панель инструментов «*Формы*» (рис. 4).



Рис. 4. Панель инструментов «*Формы*»

С помощью данной панели можно разместить на листе кнопки, флажки, переключатели, списки, счетчики, полосы прокрутки и назначить им макросы. Кроме того, установив необходимые свойства элементов, можно создать рабочую форму на рабочем листе [9].

4. Можно добавить команду в пункт меню или создать свой собственный пункт меню, возможно, содержащий вложенные команды меню.

Для этого перейдите в режим настройки панелей инструментов (см. рис. 2), выберите категорию «*Новое меню*» и перетащите мышью на строку меню команду «*Новое меню*». Затем выполните его настройку, используя кнопку «*Изменить выделенный объект*» (измените имя и, если возможно, назначьте макрос).

Для добавления отдельных команд в созданный пункт меню переключитесь в категорию «*Макросы*» и перетащите в пункт меню команду «*Настраиваемая команда меню*».

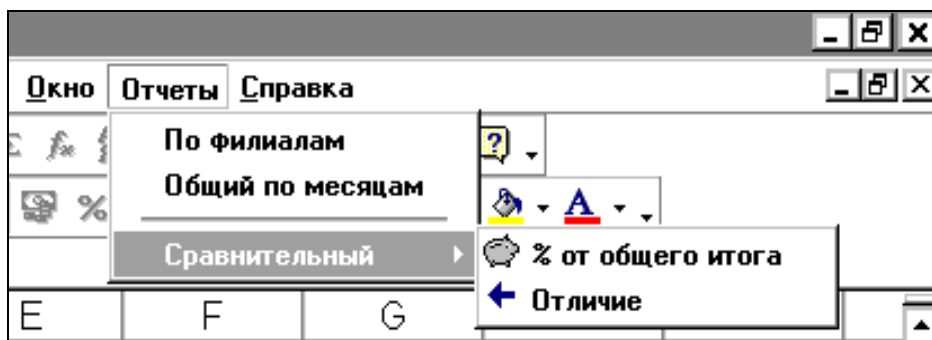


Рис. 5. Пример нового пункта меню

В примере на рис. 5 новое меню «Отчеты» содержит две команды меню: «По филиалам» и «Общий по месяцам», а также меню «Сравнительный». Данный пункт, в свою очередь, содержит еще две команды: «% от общего итога» и «Отличие».

5. Макрос можно назначить любому графическому объекту, расположенному на рабочем листе. Это может быть рисунок из стандартного набора, автофигура или сгруппированный объект.

Пример макроса

Пусть имеется таблица, содержащая сведения о сотрудниках предприятия (Код, ФИО, Дата рождения, Дата принятия, Должность, Разряд, Подразделение). Записать в автоматическом режиме макрос, выполняющий расширенный фильтр с копированием на новое место для получения списка сотрудников одного подразделения.

Так как будет выполняться фильтр с копированием результата на новое место, можно расположить область критериев и область результатов справа от таблицы. Как известно, фильтр может не выполниться если область результата занята данными. Поэтому порядок действий будет следующий:

1. Сформировать критерий фильтра (поле «подразделение»).
2. Выполнить расширенный фильтр и убедиться, что все получилось правильно.
3. Включить запись макроса.
4. Удалить предыдущий результат фильтрации (количество очищаемых строк равно количеству строк исходной таблицы).
5. Выполнить расширенный фильтр.
6. Перейти к области результата по ее имени.
7. Остановить запись макроса.
8. Изменить значение критерия и запустить макрос одним из способов.

Редактирование макросов

Макрос, записанный в автоматическом режиме, не обладает достаточной гибкостью и универсальностью. Для корректного выполнения команд должны быть заранее подготовлены соответствующие начальные условия. Самое неприятное, что действия, выполненные с помощью макроса, нельзя отменить обычной командой **Правка – Отменить**. Поэтому, если вы случайно щелкнули не на той кнопке, то в результате можете получить нежелательные изменения в на рабочих листах.

Чтобы избежать досадных недоразумений при работе с макросами и повысить эффективность их использования, необходимо научиться вносить изменения в текст макроса. Для начала познакомимся с окном редактора VBA.

Окно редактора VBA

Описание окна редактора VBA, в той или иной степени подробности, присутствует во всех изданиях, посвященных программированию в Ms Office [3, 4, 7, 9].

Редактор Visual Basic (VB-редактор) – это инструмент для создания модулей и просмотра их содержимого, создания и редактирования текста макросов, создания диалоговых окон и т.д.

VB-редактор приложения Excel аналогичен VB-редакторам других приложений, поддерживающих VBA.

Открыть окно редактора можно тремя способами.

- Выполнить команду **Сервис – Макрос – Редактор Visual Basic**.
- Нажать <Alt + F11>.
- Выполнить команду **Сервис – Макрос – Макросы**, указать название макроса и щелкнуть кнопку *«Изменить»*.

Окно редактора VBA содержит три дочерних окна, каждое из которых содержит определенную информацию о вашем проекте. Проект – это модули и другие объекты, содержащиеся в рабочей книге.

1. Окно «Project» (Проект). В окне Project отображается древовидная структура, описывающая проект: открытые в данный момент рабочие книги, листы, модули и формы, которые в них содержатся. Например, открытая рабочая книга Sample.xls содержит три рабочих листа и два листа модулей с текстами макросов.

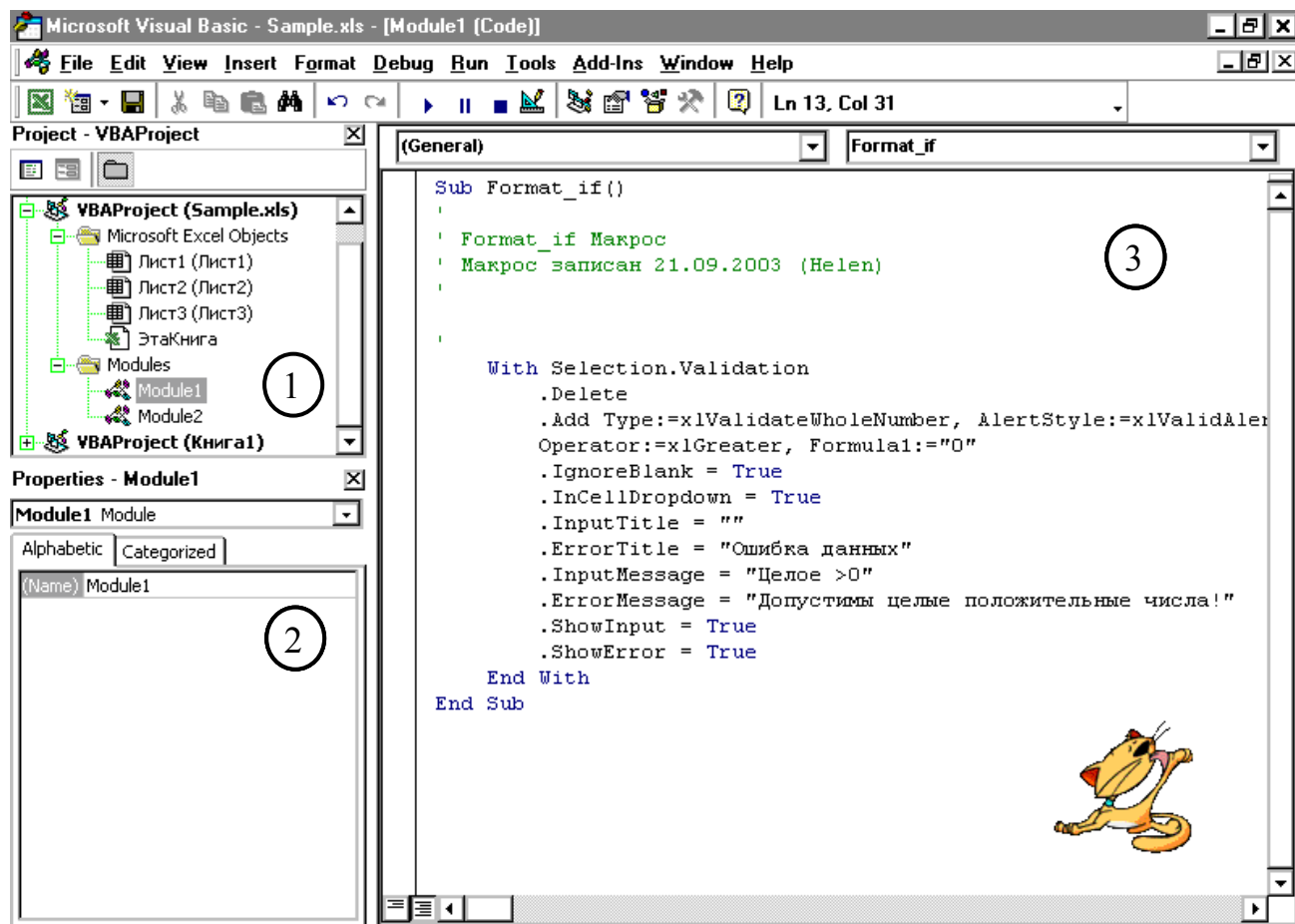


Рис. 6. Окно редактора VBA

2. Окно «Properties» (Свойства). В данном окне отображаются свойства выбранного в данный момент объекта. Более подробно с этим окном познакомимся немного позже.

3. Окно «Code» (Программа). В окне Code отображается текст макросов, содержащихся на выбранном листе модуля. Именно в этом окне мы будем читать, редактировать и создавать макросы. У верхней границы окна Code расположены два раскрывающихся списка. Список, расположенный справа, содержит перечень макросов, находящихся в данном модуле и может использоваться для быстрого перехода к нужному макросу. Окно VBA содержит собственную строку меню и набор панелей инструментов.

Команды редактора

Обычно на экране присутствует панель Standart (Стандартная). Описание всех команд занимает достаточно много места и многие из них будут непонятны на первом этапе освоения VBA. С ними вы можете познакомиться в [4, 7]. Рассмотрим основные команды, необходимые для того, чтобы начать работу в редакторе.

Основные команды редактора VB

Заккрыть окно	<ol style="list-style-type: none"> 1. Можно обычным образом, щелкнув на кнопке «Заккрыть». 2. Выполнить команду File (Файл) – Close and Return (Заккрыть и Вернуться в окно Excel)
Выделить текст	Выделение текста выполняется обычным образом с помощью мыши или клавиатуры.
Копировать	Edit (Правка) – Copy (Копировать)
Вырезать	Edit (Правка) – Cut (Вырезать)
Вставить	Edit (Правка) – Paste (Вставить)
Отменить команду	Edit (Правка) – Undo (Отменить)
Повторить	Edit (Правка) – Redo (Повторить)
Показать/спрятать окно проекта	View (Вид) – Project Explorer (Окно проекта)
Показать/спрятать окно свойств	View (Вид) – Properties Window (Окно свойств)
Вставить новый модуль	Insert (Вставка) – Module (Модуль)
Удалить модуль	<ol style="list-style-type: none"> 1. Выделите в окне «Проект» модуль, который надо удалить. 2. Выполните File (Файл) – Remove ModuleN (Удалить модуль). 3. Щелкните на кнопке «Yes», если вы решили сохранить тексты перед удалением (на всякий случай) или «No» для окончательного удаления модуля
Выполнить макрос	Run (Запуск) – Run SUB (Запуск подпрограммы)
Прервать выполнение макроса	Run (Запуск) – Break (Прервать) комбинация клавиш <Ctrl+Break>
Сбросить значения всех переменных	Run (Запуск) – Reset (Сброс)

Командам выполнения макроса (меню Run) соответствуют кнопки, показанные на рис. 7 (три кнопки слева направо): *Запуск, Прервать, Сброс*.

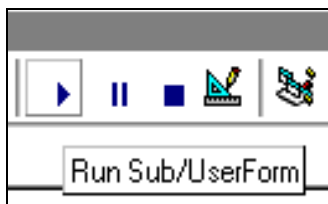


Рис. 7. Кнопки запуска и останова макроса

Изменение текста макроса

Для того, чтобы начать редактировать макрос, Вы должны сначала найти текст нужного макроса. Самый простой способ – выполнить команду **Сервис – Макрос – Макросы**, найти в списке нужный макрос и щелкнуть кнопку «*Изменить*». Откроется окно редактора VB, в окне «*Программа*» отобразится текст макроса, а в окне «*Проект*» будет выделен модуль, содержащий его. Познакомимся с текстом макроса, записанным ранее, более подробно:

```
1 Sub Фильтр()  
2 '  
3 ' Фильтр Макрос  
4 ' Макрос записан 16.12.2003 (Helen)  
5 '  
6 Range("K1").Select  
7 Selection.CurrentRegion.Select  
8 Selection.Clear  
9 Range("F2").Select  
10 Range("A1:G21").AdvancedFilter Action:=xlFilterCopy, _  
11 CriteriaRange := Range("I1:I2"), CopyToRange:=Range("K1"), Unique:=False  
12 Application.Goto Reference:="Extract"  
13 Range("K1").Select  
14 End Sub
```

Каждый VBA-макрос начинается с ключевого слова **Sub**, за которым через пробел следует имя макроса и пустые скобки.

Ключевое слово – это слово, зарезервированное в VBA для специального использования, в отличие от имени макроса, которое задается пользователем. Ключевые слова не могут быть использованы в качестве имен. Ключевые слова в редакторе обычно отображаются синим цветом.

Строки с номерами 2, 3, 4, 5 – это строки комментариев, добавленных автоматически. Каждая строка начинается с символа апостроф «'». Все, что записано после апострофа, считается комментарием.

Комментарий не является частью языка программирования и не содержит выполняемых инструкций. Комментарии служат для пояснения текста макроса. Комментарии в редакторе обычно отображаются зеленым цветом.

Иногда комментарий используют для того, чтобы отменить выполнение фрагмента или отдельных строк макроса.

При записи макроса в автоматическом режиме комментарии обычно содержат название макроса, дату и автора макроса.

Далее следует тело макроса, каждая строка которого содержит одну или несколько инструкций языка VBA. VBA-инструкции в макросе соответствуют тем действиям, которые Вы выполняли при записи макроса. Например, четвертому шагу примера макроса соответствуют строки 6, 7, 8, которые соответствуют действиям по удалению старого результата (установить рамку в таблицу результата, выделить текущую область, очистить ячейки).

Далее следуют строки 9, 10, 11, реализующие расширенный фильтр (установить рамку в исходную таблицу и выполнить команду **Данные – Фильтр – Расширенный фильтр**). Если внимательно прочитаете строки 10 и 11, то Вы найдете в них все установленные параметры расширенного фильтра.

Далее следует переход к области по ее имени.

Завершается текст макроса обязательной инструкцией **End Sub** (конец процедуры).

Для облегчения работы с текстом макроса длинные инструкции можно переносить с одной строки на другую. Признаком переноса инструкции на другую строку является комбинация символов «пробел» и «подчеркивание» в конце строки. Она указывает VBA, что следующая строка является продолжением предыдущей (строка 11 является продолжением строки 10).

Строки макроса могут быть записаны с отступом от левого края. Различные отступы способствуют визуальному восприятию текста и выделению различных логических частей макроса. Эти отступы никак не влияют на выполнение макроса. Это лишь один из приемов, который облегчает чтение и анализ текста программы.

В качестве первого примера редактирования макроса добавим в 5 строку поясняющий текст «Фильтрация таблицы «Кадры» по полю «Подразделение»».

Всегда добавляйте в макрос комментарии, пока вы еще хорошо помните, что он делает и как, особенно если макрос достаточно большой или существует несколько подобных макросов.

Visual Basic проверяет тип строки в тот момент, когда курсор покидает ее, для того, чтобы определить, правильно ли она набрана с точки зрения синтаксиса.

Синтаксис – это правила, по которым строятся инструкции языка программирования (предложения в обычном человеческом языке).

Если строка набрана правильно, то она будет раскрашена в соответствии с указанными ранее правилами. Если строка содержит ошибку, то VBA красит всю строку в красный цвет и выводит сообщение об ошибке.

Для того, чтобы вносить более существенные изменения в тексты макросов или писать свои процедуры, необходимо познакомиться для начала с простейшими операторами языка VBA.

Основные операторы языка VBA

Оператор присваивания

Оператор присваивания выглядит, как привычный знак «=». Синтаксис:

Имя_переменной = Выражение

Имя_переменной – это обозначение, которое программист дает участку памяти для хранения в нем данных определенного типа. Переменной можно дать практически любое имя, но рекомендуется делать его описательным, близким по смыслу к содержанию. Значение переменной в процессе работы программы может изменяться.

Образование имен переменных должно подчиняться правилам:

- Имя переменной начинается с буквы.
- За буквой следует любая комбинация букв, цифр, и символов подчеркивания.
- Имя переменной не должно содержать точек, пробелов, знаков арифметических операций.
- Имя не может превышать 255 символов в длину.
- Имя не может совпадать ни с одним ключевым словом VBA, именем другой переменной в данной процедуре или именем процедуры.

Выражение – любое допустимое в VBA выражение – совокупность переменных, констант, функций, объединенных знаками арифметических и логических операций.

При выполнении оператора присваивания сначала вычисляется значение выражения, стоящего **справа** от оператора, а потом полученное значение присваивается переменной, указанной слева от оператора.

При записи выражения могут использоваться следующие группы операций:

— Арифметические операции. Для арифметических операций приняты обозначения:

+ сложение; – вычитание;

* умножение; / деление;

^ возведение в степень;

\ целочисленное деление (отбрасывает дробную часть);

Mod деление по модулю. **N1 Mod N2** делит N1 на N2, возвращая остаток от деления.

— Операции сравнения. Операции сравнения иногда называют *операторы отношения*. Их применяют для формирования условия выполнения некоторой последовательности операций. Результат выполнения оператора сравнения принимает только два значения: **True** (истина) и **False** (ложь). Принятые обозначения:

= равно; <> не равно;

< меньше; > больше;

<= меньше или равно; >= больше или равно.

— Конкатенация строк. Основной знак для операции слияния (конкатенация) текстовых строк – амперсанд &.

— Логические операции. Логические операции используются при формировании условий для записи логических выражений.

AND – логическое «И». Результат принимает значения «истина», если истинны оба операнда.

OR – логическое «ИЛИ». Результат принимает значение «истина», если хотя бы один из операндов имеет значение «истина».

NOT – отрицание. Значение операнда меняется на противоположное.

Условные операторы

Существует несколько вариантов записи оператора If. Синтаксис 1:

```
If Логич_выражение Then Оператор
```

Вычисляется значение логического выражения, если оно принимает значение «истина», то выполняется Оператор. В противном случае сразу выполняется инструкция в следующей строке. Синтаксис 2 (If...Then...Else):

```
If Логич_выражение Then  
    Операторы_1  
Else  
    Операторы_2  
End If
```

Вычисляется значение логического выражения, если оно принимает значение «истина», то выполняется группа Операторы_1. В противном случае выполняется группа Операторы_2. Указанные группы операторов могут содержать внутри себя другие операторы If.

Для последовательной проверки нескольких условий удобно использовать инструкцию If...Elseif. Синтаксис 3:

```
If Логич_выражение_1 Then  
    Операторы  
Elseif Логич_выражение_2 Then  
    Операторы  
.....  
Else  
    Операторы  
End If
```

Если первое логическое выражение истинно, то выполняется соответствующая ему группа операторов после Then, иначе проверяется второе логическое выражение и, если оно истинно, выполняется соответствующая ему группа операторов и т.д. Если ни одно из логических выражений не примет значение «истина», то выполняется группа операторов после Else.

Оператор Select Case

Для выбора из большого количества ветвей процедуры можно использовать инструкцию Select Case. Простейший вариант записи инструкции:

```

Select Case Выражение
    Case Значение_выражения_1
        Операторы_1
    Case Значение_выражения_2
        Операторы_2
    .....
    Case Else
        Операторы_Else
End Select

```

Вычисляется значение выражения, в частности, выражение может быть просто именем переменной. Затем выполняется группа операторов, соответствующая полученному значению. Если значение выражения отличается от указанных в **Case**, то выполняется группа операторов, соответствующая **Case Else**.

Всегда выполняется только один из разделов **Case**.

Возможны более сложные варианты записи **Case**, когда вместо одного значения переменной указывается список значений, диапазон значений, операции сравнения (например, **Case Is >100**).

Циклические операторы

Одним из недостатков записанных макросов является их неспособность повторять операции, если вы не повторили их вручную при записи макроса.

Программные структуры, дающие возможность повторения одной или нескольких инструкций, называются *циклами*.

Некоторые циклические структуры созданы так, что они повторяются заданное количество раз. Другие позволяют организовать повторение до тех пор, пока выполняется некоторое условие.

Операторы, записанные между инструкциями начала и конца цикла, называются *телом цикла*.

Тело цикла может содержать любые операторы, в том числе и операторы цикла. Такие циклы называются *вложенными*.

Цикл **For...Next**

Данный цикл следует использовать, если требуется организовать повторение операций заданное число раз.

For переменная_цикла = нач_значение To конечн_значение [Step шаг]

Операторы

Next переменная_цикла

Переменная_цикла – любая числовая переменная (чаще всего переменная, принимающая целочисленные значения).

Step – шаг изменения значения переменной цикла. Может принимать положительные и отрицательные значения, по умолчанию имеет значение, равное единице, и может не указываться.

При выполнении цикла переменной цикла присваивается начальное значение. Затем выполняются все инструкции тела цикла. Далее значение переменной цикла изменяется на величину шага и полученное значение сравнивается с конечным значением переменной. Если конечное значение не достигнуто, то вновь выполняются все инструкции тела цикла. Когда значение переменной цикла превысит конечное значение, то управление передается инструкции, следующей за ключевым словом Next.

Самый простой пример использования данного цикла – вычисление суммы чисел по принципу накопления суммы в некоторой переменной. Рассмотрим фрагмент кода процедуры:

‘ Обозначим через S вычисляемое значение суммы

S = 0 ‘ Начальное значение суммы

For I=1 TO 150

S = S+I

Next I

В данном примере вычисляется сумма чисел от 1 до 150 и результат вычисления сохраняется в переменной S. Вычисление происходит следующим образом: первоначально значение суммы равно нулю. Затем начинает работать цикл. Переменная цикла принимает значение 1 и оно добавляется к старому значению суммы, а результат записывается снова в переменную S (строка 4).

Далее переменная цикла принимает следующее значение, равное двум, и операция повторяется, т.е. сумма как бы «накапливается» в переменной S.

Данный пример очень легко изменить для вычисления суммы четных (или нечетных) чисел. Для этого необходимо заголовок цикла переписать в виде:

For I = 2 To 150 Step 2 для суммирования четных чисел;

или

For I = 1 To 150 Step 2 для суммирования нечетных чисел.

Цикл с проверкой условия

Циклической структурой, позволяющей реализовать проверку перед началом выполнения тела цикла, является оператор Do While.

Do While логическое_выражение

Операторы

Loop

Логическое_выражение – это допустимое логическое выражение, которое задает условие, при котором будет выполняться тело цикла. Данный цикл работает, пока значение выражения истинно. Когда логическое выражение примет значение False, управление передается оператору, следующему за ключевым словом Loop, т.е. цикл завершается. В качестве логического выражения можно использовать логическую переменную, принимающую значения True или False.

Оператор Do While позволяет создавать циклы, повторяющиеся заранее неизвестное число раз.

Принудительное завершение циклов

Существует много причин, по которым может потребоваться немедленное завершение работы цикла. Например, пользователь отказался от ввода данных, решение найдено и нет смысла в продолжении вычислений.

Для немедленного завершения цикла For...Next используется инструкция Exit For. Для немедленного завершения цикла Do используется инструкция Exit Do.

Описание типа переменных

При написании программ на VBA не обязательно объявлять типы используемых переменных до их первого использования. Когда в программе встречается оператор присваивания, VBA создаст переменную, тип которой совместим с типом выражения. Такой способ создания переменной называется *неявным объявлением переменной*.

Однако, чтобы уменьшить объем занимаемой памяти и избежать ошибок, связанных с преобразованием типа данных или опечаткой в имени переменной, тип переменной лучше объявить до ее использования. Описание типа переменных выполняется с помощью оператора Dim. Общий вид записи:

Dim переменная As тип[, переменная As тип,]

Возможные типы данных приведены в табл. 2.

Таблица 2

Имя типа данных	Размер в байтах	Описание
Byte	1	Положительные числа от 0 до 255
Boolean	2	Логические значения True или False
Date	8	Комбинация даты и времени. Дата от 1.01.100 до 31.12.9999. Время от 00:00:00 до 23:59:59
Double	8	Отрицательные от -1.79×10^{308} до -4.9×10^{-324} Положительные от 4.9×10^{-324} до 1.79×10^{308}
Integer	2	Числа от -32768 до 32767
Long	4	Числа от -2147483648 до 2147483647
Object	4	Используется для доступа к любому объекту VBA. Хранит адрес объекта.
Single	4	Отрицательные от -3.4×10^{38} до -1.4×10^{45} Положительные от 1.4×10^{45} до 3.4×10^{38}
String	1 байт на символ	Текст. Может содержать до 2 миллионов символов.
Variant	16 байт+1 байт на символ	Служит для хранения данных любого типа

Любая переменная, для которой не указан тип данных, становится переменной типа Variant. Такие переменные требуют больше памяти и большинство операций выполняется для них медленнее, чем для других типов.

Переменная типа Variant может неправильно обрабатывать дату, полученную с листа Excel, если дата не была отформатирована на листе одним из стандартных форматов даты. Однако, она будет правильно воспринята переменной типа Date. Аналогично, при записи на рабочий лист, дата не будет воспринята правильно из переменной типа Variant.

Чтобы запретить неявное описание типа переменных в модуле, необходимо добавить в начало модуля, до первой процедуры модуля, команду Option Explicit.

В этом случае при любой попытке использовать переменную, тип которой не указан явно в операторе Dim, будет выдано сообщение об ошибке.

Функции MsgBox и Inputbox

Для организации взаимодействия с пользователем в VBA служат функции MsgBox и InputBox.

Функция MsgBox позволяет вывести на экран окно сообщения с заданной комбинацией кнопок. Общая форма записи:

MsgBox сообщение[, кнопки][, заголовок][, HelpFile][, Context]

Сообщение – текстовая константа или переменная.

Кнопки – комбинация специальных констант VBA, определяющая состав кнопок, отображаемых в окне.

Заголовок – текстовая константа или переменная, задающая строку заголовка окна.

HelpFile – строковое выражение, содержащее имя файла помощи, созданного компилятором Windows Help File.

Context – число, соответствующее тематическому разделу в файле помощи. Аргументы HelpFile и Context указываются только вместе.

Единственным обязательным аргументом данной функции является текст сообщения. Например:

MsgBox “Данные успешно записаны”

Данная инструкция отобразит на экране окно, представленное на рис. 8.

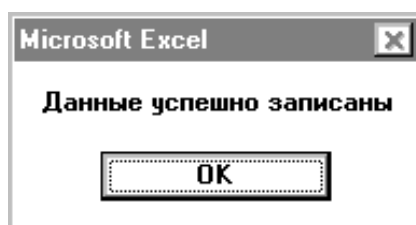


Рис. 8. Простейшее сообщение

По умолчанию окно содержит кнопку ОК и заголовок «Microsoft Excel».

Если текст сообщения достаточно длинный, то его можно разместить в разных строках. Для этого в строку сообщения включается специальная константа VBA, означающая создание новой строки – vbNewLine.

Например, строка вида:

MsgBox “Данные” & vbNewLine & “успешно” & vbNewLine & “записаны”
отобразит окно, представленное на рис. 9 (сравните с окном на рис. 8).

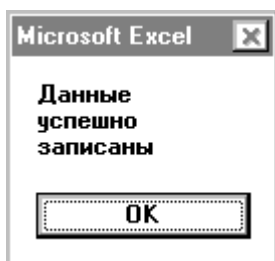


Рис. 9. Перенос текста по строкам

Кроме того, константа vbTab (табуляция) соответствует нажатию клавиши <Tab> на клавиатуре. Этот символ можно включать в строки для выравнивания текста или размещения его по колонкам. Например:

```
Message = "Первая" & vbTab & "строка" & vbTab & "данных" & vbNewLine
Message = Message & "Еще" & vbTab & "строка" & vbTab & "данных"
MsgBox Message
```

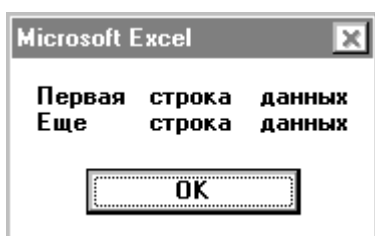


Рис. 10. Использование табуляции





Используя аргументы «кнопки» и «заголовок», можно сформировать более интересные окна сообщений. Рассмотрим возможные значения аргумента «кнопки». Как уже отмечалось, состав кнопок устанавливается специальными константами VBA, описание которых приведено в табл. 3.

Таблица 3

Константа	Кнопки
vbOKOnly	Только кнопка ОК
vbOKCancel	Кнопки ОК и Отмена
vbAbortRetryIgnore	Кнопки Стоп, Повтор, Пропустить
vbYesNoCancel	Кнопки Да, Нет и Отмена
vbYesNo	Кнопки Да, Нет
vbRetryCancel	Кнопки Повтор и Отмена

Следующая группа кнопок устанавливает вид пиктограммы, присутствующей в окне сообщения (табл. 4).

Таблица 4

Константа	Описание	Пиктограмма
vbCritical	Значок критической ошибки	
vbQuestion	Знак вопроса	
vbExclamation	Восклицательный знак	
vbInformation	Знак информационного сообщения	

Если окно сообщения содержит более одной кнопки, то пользователь может указать, какая кнопка должна быть установлена по умолчанию. Тогда при нажатии «Enter» будет выполняться действие, предусмотренное вами для данной кнопки. Соответствующие константы указаны в табл. 5.

Таблица 5

Константа	Описание
VbDefaultButton1	По умолчанию выделена первая кнопка
VbDefaultButton2	По умолчанию выделена вторая кнопка
VbDefaultButton3	По умолчанию выделена третья кнопка
VbDefaultButton4	По умолчанию выделена четвертая кнопка

Общий вид окна формируется как сумма констант, причем не обязательно должны присутствовать константы каждой группы. Например, на рис. 11 представлены различные варианты окон сообщений, соответствующие приведенным ниже инструкциям.

```
Sub sample_1()
```

```
    MsgBox "Фильтровать?", vbOKCancel + vbQuestion + vbDefaultButton2, _  
        "Отбор данных"
```

```
    MsgBox "Отбор данных произведен", vbOKOnly + vbInformation, _  
        "Отбор данных"
```

```
    MsgBox "Ошибка в данных!", vbOKOnly + vbCritical, "Отбор данных"
```

```
    MsgBox "Ошибка в данных!", vbCritical + vbAbortRetryIgnore, _  
        "Отбор данных"
```

```
End Sub
```

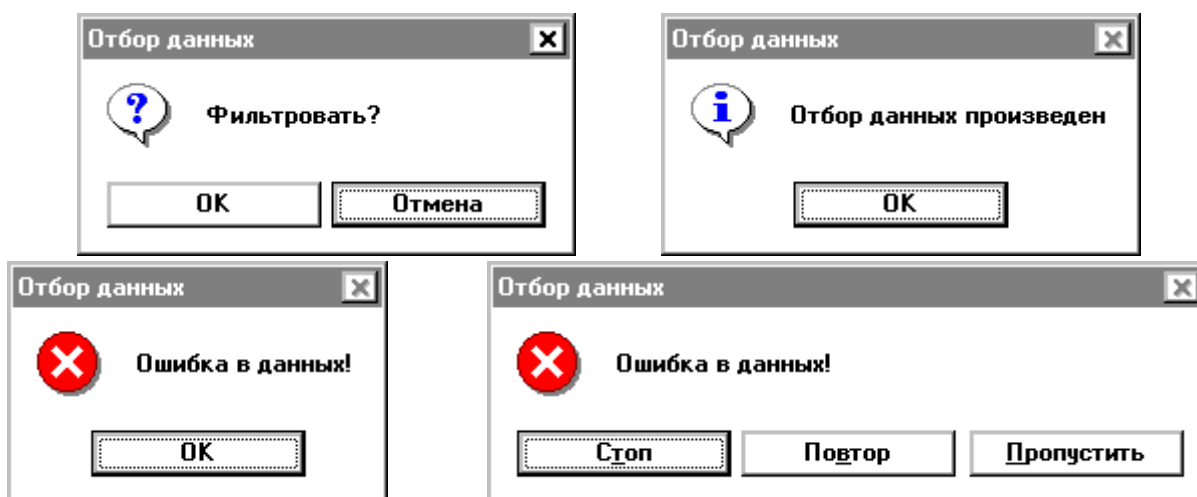


Рис. 11. Примеры окон сообщений

Для того, чтобы проверить, какую кнопку выбрал пользователь и выполнить соответствующие действия, необходимо проверить код кнопки, возвращаемый функцией `MsgBox`. В этом случае обращение к функции должно быть записано в виде:

Переменная = `MsgBox`(сообщение, кнопки, заголовок)

Значения, возвращаемые функцией `MsgBox`, представлены в табл. 6.

Таблица 6

Константа	Значение	Кнопка	Константа	Значение	Кнопка
<code>vbOK</code>	1	OK	<code>vbIgnore</code>	5	Пропустить
<code>vbCancel</code>	2	Отмена	<code>vbYes</code>	6	Да
<code>vbAbort</code>	3	Стоп	<code>vbNo</code>	7	Нет
<code>vbRetry</code>	4	Повтор			

Например, чтобы завершить выполнение макроса, если пользователь выбрал кнопку «Отмена», можно записать инструкцию:

`If answer = 2 Then Exit Sub`

или

`If answer = vbCancel Then Exit Sub,`

где `answer` – значение, возвращенное `MsgBox`.

При указании файла помощи в окно необходимо добавить кнопку Справка, что выполняется добавлением в аргумент «кнопки» константы `vbMsgBoxHelpButton`.

Для организации ввода данных в режиме диалога с пользователем используется функция `InputBox`, отображающая на экране окно, всегда содержащее кнопки `OK` и `Cancel` и поле для ввода данных. Общий вид записи:

```
s = InputBox(сообщение[,заголовок][,знач_по_умолч][,X][,Y][,HelpFile,Context)
```

Сообщение – символьная константа или переменная длиной до 230 символов.

Заголовок – символьная константа или переменная, содержащая текст заголовка окна.

Знач_по_умолч – символьная константа или переменная, задающая текст, отображаемый в поле ввода по умолчанию.

X – число, задающее смещение окна диалога от левого края экрана.

Y – число, задающее смещение окна диалога от верхнего края экрана.

Значения X и Y измеряются в *twip*ах, один *twip* приблизительно равен 0.0007 дюйма.

Значения необязательных аргументов `HelpFile` и `Context` аналогичны аргументам функции `MsgBox`. При определении файла помощи в окно автоматически будет добавлена кнопка «Справка».

Например, инструкция

```
S = InputBox("Введите Ваше Имя", "Регистрация")
```

отобразит на экране окно, показанное на рис. 12.

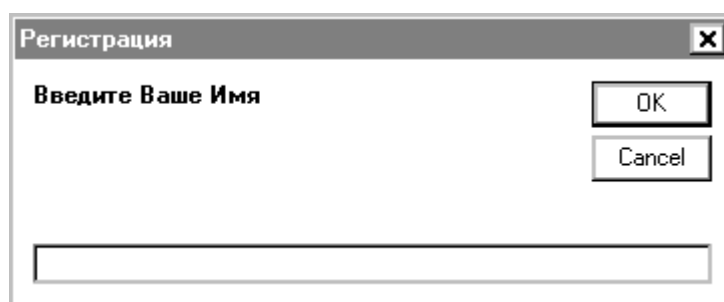


Рис. 12. Пример окна ввода

Если включить в список аргументов значение по умолчанию, то оно будет присутствовать в поле ввода при отображении окна (см. рис. 13).

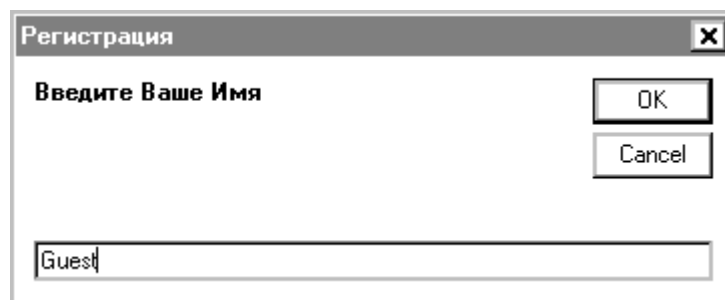


Рис. 13. Поле ввода содержит значение, установленное по умолчанию

Переменная *S* будет содержать текстовое значение, введенное пользователем. Если пользователь выбрал кнопку *Cancel*, то функция возвращает строку нулевой длины. В данном случае завершение работы макроса может быть записано в виде:

```
If S = "" Then Exit Sub
```

Обратите внимание – внутри кавычек пробела нет! Так записывается строка нулевой длины.

Конечно, макрос может и не завершаться при нажатии кнопки *Cancel*, все зависит от алгоритма работы вашей процедуры.

При анализе ответа пользователя следует помнить, что сравнение текстовых строк выполняется посимвольно, т.е. строки «Пример» и «пример» – это две *разные* строки, т.к. первые символы введены в разных регистрах. В этом случае можно воспользоваться встроенными функциями VBA для работы со строками. Перечень функций VBA, сгруппированных по категориям, можно получить, если открыть окно просмотра объектов в редакторе VBA, нажав клавишу F2. Далее в списке «Project/Library» выберите «VBA». В списке «Classes» показаны различные категории объектов, функций, процедур и констант, определенных VBA. Найдите нужную категорию функций (Conversion (преобразование типа), DateTime (дата и время), Information (информационные), Math (математические) или Strings (текстовые). Список функций указанной категории отобразится в соседнем поле. Щелкните на кнопке со знаком вопрос, чтобы получить справку по выделенной функции. Краткая информация о функции отображается внизу окна. Описание некоторых функций приведено в [4].

Форматирование значений данных

Для того, чтобы получить любой формат отображения данных, преобразовать числа или даты в строки или отформатировать строки в соответствии с вашими требованиями, можно использовать функцию *Format*. Функция *Format*

использует те же самые символы формата, что и Excel (откройте окно **Формат – Ячейки / Число** и в списке «*Числовые форматы*» укажите «*все форматы*»). Обращение к функции:

Format(переменная, “строка формата”)

Для создания пользовательских форматов данных используются символы, приведенные в табл. 7.

Таблица 7

Символ	Описание
0	Резервирует место для цифры, отображается либо цифра, либо ноль
#	Резервирует место для числового разряда, отображается либо цифра, либо ничего
.	Точка разделяет целую и дробную части числа
%	Устанавливает процентное отображение числа, аргумент при этом умножается на 100
d	День месяца без лидирующего нуля
dd	День месяца с лидирующим нулем
ddd	Краткое название дня недели
dddd	Полное название дня недели
w	Номер дня недели (воскресенье – 1)
ww	Неделя года в виде целого числа от 1 до 54
m	Номер месяца без лидирующего нуля
mm	Номер месяца с лидирующим нулем
mmm	Краткое название месяца
mmmm	Полное название месяца
y	День года
yy	Год в виде двух цифр
yyyy	Полное отображение года
/	Разделитель значений дня, месяца и года в формате даты
q	Номер квартала
h или hh	Отображает часы

Символ	Описание
n или nn	Минуты
s или ss	Секунды
:	Двоеточие используется как разделитель в формате времени
@	Отображает символ или пробел
&	Отображает символ или ничего
>	Отображает строку в верхнем регистре
<	Отображает строку в нижнем регистре

Например, отображение текущей даты может быть записано по-разному:

```
Sub sample2()
```

```
    MsgBox "Сегодня " & Format(Date, "dd.mm.yy")
```

```
    MsgBox "Сегодня " & Format(Date, "dddd, dd mmmm, yyyy")
```

```
End Sub
```

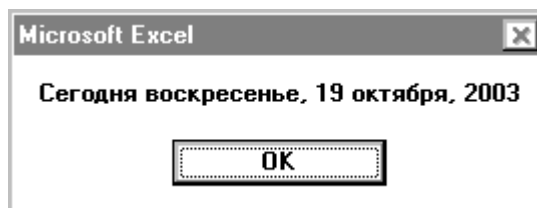
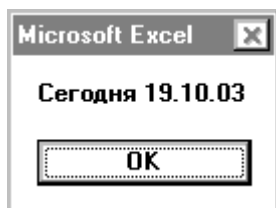


Рис. 14. Использование форматов даты

Используя числовые форматы, можно изменять вид отображения числа. Различные варианты показаны на рис. 15.

```
Sub sample3()
```

```
    p = 123.4506
```

```
    MsgBox Format(p, "00000.000")
```

```
    MsgBox Format(p, "# ##0.00")
```

```
    MsgBox Format(p, "#0.00%")
```

```
End Sub
```

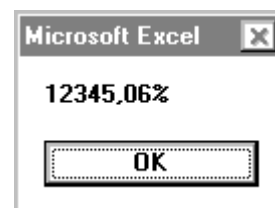
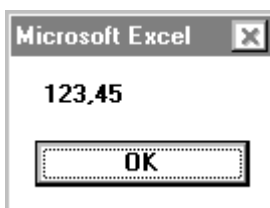
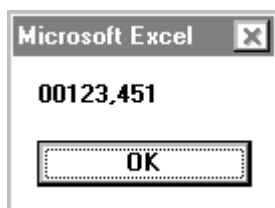


Рис. 15. Использование числовых форматов

Функции пользователя

В Excel имеется множество встроенных функций рабочего листа, доступ к которым осуществляется через Мастер функций. Однако среди них можно не найти подходящей для своих вычислений, либо для получения результата необходимо записать последовательность вычислений с использованием различных функций. Если такие действия приходится выполнять многократно, то лучше оформить их в виде самостоятельной функции.

Функция пользователя (UDF, user defined function) – это записанная в модуле VBA процедура-функция. В отличие от уже знакомой процедуры Sub, функция не может быть создана с помощью средств автоматической записи. Функция, в отличие от макроса, не выполняет никаких действий над рабочим пространством Excel, она только производит операции над данными, полученными через список аргументов или извлеченными с рабочих листов.

Общая форма записи:

Function имя(список формальных аргументов)

Операторы

End Function

Имя функции подчиняется тем же правилам записи имен, что и имя переменной в VBA.

Список аргументов – это перечень имен переменных, которые будут использоваться в функции для вычислений. Аргументы записываются через запятую. Список аргументов – необязательный параметр, можно написать функцию, которая не будет их использовать. При отсутствии аргументов за именем функции следуют пустые скобки.

С помощью списка аргументов в функцию передаются значения, необходимые для выполнения вычислений. Для того, чтобы функция вернула результат вычислений, необходимо, чтобы в ней присутствовал оператор вида:

Имя = выражение

Эта конструкция называется *присваивание результата функции*, т.е. имя функции используется как переменная и ей присваивается значение. Функция может иметь несколько различных операторов такого вида, например, при использовании разветвляющегося вычислительного процесса. Обратите внимание, что VBA не выдаст сообщений об ошибке, если вы забыли записать такой оператор.

Рассмотрим простой пример создания и использования функции.

Вычислить сумму налога на имущество по следующей схеме: если стоимость имущества меньше 850 минимальных зарплат (мин. зар.), то налога нет, если до 1700 мин. зар., то 5% от суммы, превышающей 850 мин. зар., иначе 10% от суммы, превышающей 850 мин. зар. (пример учебный).

Исходя из условия задачи, аргументами функции будут: стоимость имущества, размер минимальной зарплаты. На листе модуля запишем код:

```
Function nalog(cost, salary_min)
'вычисление налога на имущество
    k = cost / salary_min
    If k <= 850 Then
        nalog = 0
    ElseIf k <= 1700 Then
        nalog = (cost - 850 * salary_min) * 0.05
    Else
        nalog = (cost - 850 * salary_min) * 0.1
    End If
End Function
```

На одном листе модуля может быть записано несколько функций. Функция, созданная пользователем на листе модуля, доступна на рабочем листе в Мастере функций в категории «Определенные пользователем». При обращении к функции пользователя на экране отображается окно, аналогичное по своему виду окну любой стандартной функции. Пользователь задает значения необходимых аргументов, используя константы или адреса ячеек и диапазонов рабочего листа, содержащие исходные данные. Для вычислений доступны функции открытых рабочих книг, иначе в ячейке будет выдано сообщение об ошибке.

nalog

Cost B1 = 850000

Salary_min B2 = 600

= 17000

Для получения сведений о функции и ее аргументах нажмите кнопку "Справка".

Cost

Значение: 17000

OK Отмена

Рис. 16. Обращение к функции

Как видно на рис. 16, аргументы функции отображаются так, как они записаны в функции. Для облегчения понимания функции вы можете давать имена переменным по-русски. Краткий поясняющий текст, который будет отображаться при вызове функции, можно записать в окне свойств функции. Для этого, находясь на листе модуля, нажмите F2, откроется окно просмотра объектов. В списке библиотек выберите «VBA Project», выделите имя функции и в контекстном меню укажите пункт «Properties». Заполните поле «Descriptions».

У многих встроенных функций Excel, кроме *обязательных* аргументов, которые должны указываться при каждом обращении к функции, имеются так называемые *необязательные аргументы*. В окне ввода аргументов функции их имена отображаются серым цветом (например, для функции ППЛАТ это аргументы БС и ТИП). Если значение такого аргумента не указывается, то в вычислениях используется значение, установленное по умолчанию.

Для того, чтобы установить для аргумента признак «необязательный», необходимо перед его именем в списке аргументов записать ключевое слово **Optional**. В списке аргументов функции сначала перечисляются все обязательные аргументы, а затем – необязательные. Ключевое слово **Optional** записывается перед каждым аргументом.

В тексте функции до использования необязательного аргумента надо проверить, был ли он задан. Чтобы определить, указано значение аргумента, или нет, используется функция **IsMissing(аргумент)**. Данная функция возвращает значение «Истина», если аргумент **не указан**, т.е. должно использоваться значение по умолчанию. Например, в рассмотренном примере можно ввести необязательные аргументы для значений кратности минимальным зарплатам и процентных ставок. Для примера введем необязательные аргументы для процентных ставок. По умолчанию будут использованы значения, указанные в условии задачи. Внесем необходимые изменения в код (выделены в тексте):

```
Function nalog(cost, salary_min, Optional rate_min, Optional rate_max)
    If IsMissing(rate_min) Then rate_min = 0.05
    If IsMissing(rate_max) Then rate_max = 0.1
    k = cost / salary_min
    If k <= 850 Then
        nalog = 0
    ElseIf k <= 1700 Then
        nalog = (cost – 850 * salary_min) * rate_min
    Else
        nalog = (cost – 850 * salary_min) * rate_max
    End If
End Function
```

Функция `IsMissing` предназначена только для проверки необязательных аргументов типа `Variant`. Для необязательных аргументов любого другого типа VBA выполняет инициализацию значений как для новых переменных и функция `IsMissing` всегда возвращает значение `False`. В этом случае необходимо выполнять непосредственную проверку значения аргумента.

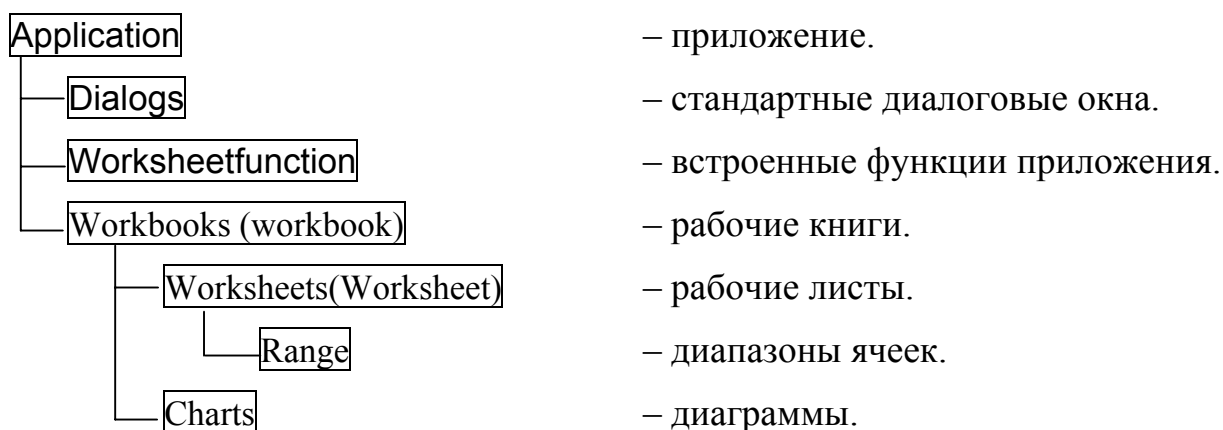
ОБЪЕКТЫ, СВОЙСТВА, МЕТОДЫ

VBA является объектно-ориентированным языком программирования. Он может оперировать не только отдельными переменными, но и объектами рабочего пространства Excel. Объектная модель Excel содержит более 100 встроенных объектов и несколько объектов, совместно используемых всеми приложениями Office.

Прежде, чем создавать приложение для Excel, необходимо представлять себе функционирование различных объектов и их иерархию.

Полная схема иерархии объектов достаточно сложна и приведена в справочной системе. Объектная модель Excel содержит более 100 встроенных объектов и несколько объектов, совместно используемых всеми приложениями Office. Познакомимся с некоторыми наиболее часто используемыми объектами.

Объектом самого верхнего уровня является `Application` – приложение Excel. Следующий уровень – `Workbooks` – рабочие книги. Далее следуют `Sheets` – листы рабочей книги и диапазоны ячеек – `Range`.



Все объекты Excel имеют **свойства** и **методы**, а VBA является инструментом, посредством которого можно управлять объектами, используя их свойства и методы. Любую реальную вещь можно рассматривать как объект. Каждая вещь может быть описана с помощью прилагательных, которые разносторонне описывают данный объект. Например, стол может быть деревянный, полированный, светло-коричневый, квадратный, сторона 80 см, высота 75 см и

т.д. Если перечислить все **свойства** объекта, то получится его точное и полное описание.

Методы характеризуют действия, которые можно выполнить над объектом или с помощью объекта. Например, лампа – включить, выключить, пере-
ставить.

Каждый объект Excel также имеет уникальный набор свойств, которые служат для описания объекта и уникальный набор методов, представляющих допустимые действия над объектом или с его помощью.

Каким образом иерархия объектов влияет на написание кода? Чтобы управлять свойствами и методами объекта, необходимо знать, как записать ссылку на данный объект. Для этого нужно спуститься вниз от корня дерева и записать имена родительских объектов, отделяя их друг от друга **точкой**.

Пример:

```
Application.Workbooks("a.xls").Sheets("Sklad").Range("A1:D8").Value = 1
```

Приведенная строка кода означает: «Записать значение 1 в диапазон ячеек A1:D8, расположенный на листе Sklad открытой рабочей книги a.xls».

Во многих случаях нет необходимости перечислять все ступени иерархии, все зависит от контекста в котором выполняются эти действия.

Для ссылок на текущие активные объекты можно использовать короткие имена:

ActiveWorkbook	– текущая рабочая книга;
ActiveSheet	– текущий рабочий лист;
ActiveCell	– текущая ячейка рабочего листа;
Selection	– выделенная область рабочего листа.

В этом случае, если известно, что в данный момент активен лист Sklad рабочей книги a.xls, данный оператор можно записать в виде:

```
Range("A1:D8").Value = 1
```

Используя VBA, можно выполнять два типа действий со свойствами.

1. Можно **получить** значение свойства, т.е. определить, какое значение оно имеет в данный момент.
2. Можно **устанавливать** свойство, т.е. изменять его значение (Например, изменить шрифт и цвет заголовка).

Для выполнения этих действий необходимо указать имя объекта и название свойства.

Таблица 8

Свойства объекта Application

Свойство	Значение	Действия
Caption	Надпись, отображаемая в заголовке. Строка знаков	Чтение/запись
ScreenUpdating	Если установить True, то экран обновляется при выполнении подпрограммы, если False, то Excel ждет окончания выполнения подпрограммы для обновления экрана.	Чтение/запись
DisplayStatusbar	Отобразить/спрятать строку состояния	Чтение/запись
Displayformulabar	Отобразить/спрятать строку формул	Чтение/запись
DisplayAlerts	True – отображать встроенные предупреждения о работе программы. False – не отображать встроенные предупреждения	Чтение/запись
WindowState	Устанавливает размер окна. Возможные значения: xlMaximized (максимальный), xlMinimized (минимальный), xlNormal – обычный.	Чтение/запись
StatusBar	Выводит заданный текст в строку состояния	Чтение/запись

Таблица 9

Методы объекта Application

Метод	Значение
Calculate	Перевычислить все формулы на всех рабочих листах открытых книг.
Quit	Закрывает Excel.
Run	Запустить на выполнение макрос.
OnTime	Назначает выполнение процедуры на определенное время.

Примеры:

Изменить заголовок окна приложения:

Application.Caption = "Личный бюджет"

Отменить обновление экрана во время выполнения макроса:

Application.ScreenUpdating = False

Развернуть окно приложения на весь экран:

`Application.WindowState = xlMaximized`

Отменить отображение строки формул:

`Application.DisplayFormulaBar = True`

Вывести текст в строку состояния:

`Application.DisplayStatusBar = True`

`Application.StatusBar = "Обработка данных...Пожалуйста, подождите..."`

Завершить работу приложения

`Application.Quit`

Объект Workbook

Объект Workbook является элементом семейства Workbooks и представляет файл рабочей книги. Свойства и методы данного объекта позволяют выполнять операции над файлами (табл. 10, табл. 11).

Таблица 10

Свойства объекта Workbook и семейства Workbooks

Свойство	Значение	Действия
Name	Имя рабочей книги	Чтение
Saved	Если true, то со времени последнего сохранения никаких изменений в книге не сделано, если False то изменения были	Чтение/запись
Count	Количество объектов в семействе Workbooks	Чтение
Path	Путь к каталогу, из которого открыта книга	Чтение

Таблица 11

Методы объекта Workbook и семейства Workbooks

Метод	Действие
Add	Добавляет новую рабочую книгу
Activate	Активизирует первое окно, связанное с данной рабочей книгой и делает эту книгу активной
Close	Закрывает рабочую книгу
Save	Сохраняет рабочую книгу. Аргументов нет
SaveAs	Сохраняет книгу в другом файле с новым именем
Open	Открывает существующую книгу.

Примеры:

Определить имя текущей рабочей книги:

```
S = Application.ActiveWorkbook.Name
```

Отобразить путь активной книги:

```
S = ActiveWorkbook.Path
```

Сохранить рабочую книгу:

```
Workbooks("primer.xls").Save
```

Закрыть рабочую книгу:

```
Workbooks("primer.xls").Close
```

Метод `Close` имеет три необязательных аргумента: сохранять изменения, имя файла и признак рассылки по электронной почте. Если опустить первый аргумент (как в приведенном примере), то будет выдан запрос на подтверждение изменений.

При указании аргументов можно использовать два способа: «позиционный», когда аргументы должны быть заданы в определенной последовательности и «по имени», когда указывается точное название аргумента.

Позиционный способ:

```
Workbooks("Primer.xls").Close True, "primer2.xls", False
```

По умолчанию значение второго аргумента – текущая книга, а третьего – `False`.

Указание аргументов по имени:

```
Workbooks("Primer.xls").Close SaveChanges:=True
```

Обратите внимание, что значение аргументу метода присваивается операцией присваивания «двоеточие равно – `:=`», а не просто знак равенства. При указании аргументов по имени они могут следовать в любом порядке.

Объект `Worksheet` и семейство `Worksheets`

Рабочие листы обеспечивают мощную основу для обработки данных и их отображения. Кроме того, можно использовать около 400 встроенных функций рабочего листа.

Для обращения к листам рабочей книги можно также использовать семейство `Sheets`. Оно несколько шире, чем `Worksheets`, т.к. может содержать, например, листы диаграмм.

Таблица 12

Свойства объекта Worksheet и семейства Worksheets

Свойство	Значение	Действия
Count	Количество листов в рабочей книге	Чтение
Name	Имя листа	Чтение/запись
Visible	Если True, то лист отображается на экране, если False – лист скрыт	Чтение/запись

Таблица 13

Методы объекта Worksheet и семейства Worksheets

Метод	Действие
Activate	Активизирует указанный рабочий лист
Delete	Удаляет рабочий лист
Add	Добавляет новый рабочий лист
Calculate	Перевычислить формулы указанного листа

Примеры:

Переименовать указанный рабочий лист:

`Worksheets("Лист1").Name = "Январь"`

Переименовать текущий лист:

`ActiveSheet.Name = "Февраль"`

Активизировать лист с именем "Лист1":

`Sheets("Лист1").Activate`

Удалить лист с именем "Итоги":

`Sheets("Итоги").Delete`

Способы доступа к ячейкам. Объекты Range и Cells

Следующим объектом иерархии после рабочего листа является объект Range (диапазон). Объект Range попадает в промежуточную область между единичным объектом и семейством. Он используется для ссылок на ячейку или диапазон ячеек на рабочем листе. Это, по определению, единичный объект, однако у него есть черты, присущие семействам. Например, обращение к ячейке по адресу или имени.

Ячейка является частным случаем диапазона, состоящим из одной ячейки. Поэтому объект Range позволяет работать также и с одной ячейкой. Описание некоторых свойств и методов приведено в табл. 14 и табл. 15.

Таблица 14

Свойства объекта Range

Свойство	Значение	Действия
Value	Значение ячейки	Чтение/запись
Name	Имя диапазона ячеек	Чтение/запись
Formula	Представленная в виде строки знаков формула, которая содержится в диапазоне, включая знак =	Чтение/запись
Count	Количество ячеек в диапазоне	Чтение
CurrentRegion	Текущий диапазон, ограниченный пустыми строками и столбцами, содержащий указанную ячейку	Чтение
Text	Содержание диапазона в текстовом формате	Чтение/запись
Columns	Набор столбцов указанного диапазона. Имеет свойство Count, возвращающее количество столбцов в наборе	Чтение
Rows	Набор строк указанного диапазона. Имеет свойство Count, возвращающее количество строк в наборе	Чтение

Таблица 15

Методы объекта Range

Метод	Действие
Calculate	Перевычисляет все формулы диапазона
ClearContents	Очищает все значения, но оставляет форматирование ячеек
Clear	Полностью очищает ячейку
Copy	Копирует диапазон в другой диапазон или в буфер обмена. Если аргумент destination не задан, то копирование производится в буфер обмена.
Insert	Вставка ячейки, диапазона, строки или столбца
Select	Выделение диапазона
Cut	Копирует диапазон в другой диапазон или в буфер обмена с удалением. Если аргумент destination не задан, то копирование с удалением производится в буфер обмена.

Примеры:

Записать в диапазон ячеек текущего листа значение 5:

```
ActiveSheet.Range("A1:A10").Value = 5
```

Записать в ячейку B2 значение «Курс \$»

```
ActiveSheet.Range("B2").Value = «Курс $»
```

Присвоить ячейке C2 имя «Курс \$»:

```
Activeshet.Range("C2").Name="Курс $"
```

Записать в диапазон ячеек формулу:

```
Sheets("Отчет").Range("B2:B10").Formula = "=A2+A2*20%"
```

Формула записывается в диапазон ячеек, при этом в тексте инструкции указана формула для первой ячейки диапазона в относительных адресах, в остальных ячейках адреса автоматически изменятся, т.е. в ячейке B3 появится формула =A3+A3*20%. Следующий пример демонстрирует запись формулы с использованием абсолютного адреса:

```
Sheets("Отчет").Range("B2:B10").Formula = "=A2*$C$2"
```

Выделить диапазон A1:D10 на листе «Отчет»:

```
Sheets("Отчет").Range("A1:D10").Select
```

Удалить содержимое ячеек диапазона:

```
Sheets("Отчет").Range("A1:D10").ClearContents
```

Полностью очистить диапазон:

```
Sheets("Отчет").Range("A1:D10").Clear
```

Выделить текущий диапазон

```
Sheets("Отчет").Range("A1").CurrentRegion.Select
```

Определить количество строк в текущем диапазоне, содержащем ячейку A1:

```
k = Sheets("Отчет").Range("A1").CurrentRegion.Rows.Count
```

Копировать диапазон ячеек на текущий лист, начиная с E1:

```
ActiveSheet.Range("A1:A10").Copy destination:=ActiveSheet.Range("E1")
```

Объект Cells – это альтернативный способ обращения к ячейке, удобный в том случае, когда номер строки или столбца заранее не известен, либо, когда необходимо перебрать все ячейки из диапазона. Обращение к ячейке записывается в виде: Cells(номер_строки , номер_столбца). Например:

Cells(1,1) – ячейка A1;

Cells(8,2) – ячейка B8.

Представляет интерес совместное использование Range и Cells, т.к. многие свойства и методы применимы только к объекту Range. Например, для указания диапазона ячеек можно использовать запись

`Range(Cells(1,1) , Cells(8,1))`, которая соответствует `Range("A1:A8")`.

В качестве номера строки или столбца можно использовать имена переменных. В этом случае мы имеем возможность указать любой, заранее не известный диапазон на рабочем листе. Например, запись:

`Range(Cells(1,1), Cells(n,4))`

позволит определить диапазон с заранее неизвестным количеством строк.

Стандартные диалоговые окна

Существует возможность вызова из пользовательского макроса стандартных диалоговых окон Excel, при этом они не возвращают никаких аргументов в макрос, а выполняют свои обычные функции. Диалоговые окна Excel применяются к конкретному объекту, поэтому их нельзя отобразить вне соответствующего контекста. В этом случае будет выдано сообщение об ошибке. Каждое окно именуется константами Excel, которые начинаются с префикса `xlDialog`, далее следует название окна. Например, `xlDialogSaveAs` – окно «Сохранить как...», `xlDialogPrint` – окно «Печать» и т.д.

Прежде, чем вызвать конкретное окно, необходимо активизировать соответствующий рабочий лист, указав, например, ячейку на нем. Например, следующий код последовательно отображает на экране окна «Печать» и «Открыть документ». Для отображения окна используется метод `Show`.

```
Sub Display_Dialogs()  
    Application.Dialogs(xlDialogPrint).Show  
    Application.Dialogs(xlDialogOpen).Show "*.xls"  
    Application.Dialogs(xlDialogOpen).Show "*.*"  
End Sub
```


СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ПРИЛОЖЕНИЙ

Используя VBA, можно создавать пользовательские приложения, оперирующие объектами не только Excel, но и других программ. При разработке приложения можно рассматривать несколько подходов.

1. Создание форм на рабочих листах с использованием элементов управления, доступных на панели инструментов «Формы», специального форматирования ячеек, проверки данных, защиты листов и макросов, назначенных элементам управления. Такой подход достаточно подробно рассмотрен в [9].
2. Создание приложений с использованием диалоговых окон, разработанных пользователем с учетом конкретной задачи. В данном случае можно оставить на экране формы своего приложения, спрятав от пользователя Excel. Рабочие листы с отчетами можно отображать на экране в случае необходимости [1, 3, 4, 7].
3. Можно комбинировать оба способа, вызывая, например, нужные формы с рабочего листа или из собственной строки меню.

Создание пользовательских форм

Создание пользовательских диалоговых окон происходит в редакторе VBA. Новая форма добавляется в проект командой Вставить – UserForm (Insert – Userform). Форма служит базой пользовательского диалогового окна, на которой размещают необходимые элементы управления.

Основные элементы управления, доступные разработчику, размещены на панели инструментов «Элементы управления», которая появляется, если нажата кнопка  (ToolBox) на панели инструментов редактора.

Любое окно Windows строится с использованием данного набора элементов. Названия элементов и номера соответствующих им кнопок приведены в табл. 16 и на рис. 17.

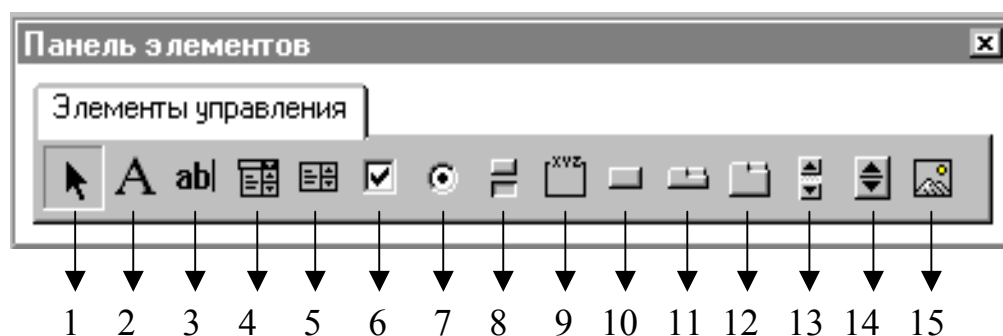


Рис. 17. Панель элементов управления

Таблица 16

Номер кнопки на рис. 16	Элемент управления	Название
2	Надпись	Label
3	Поле ввода	TextBox
4	Поле со списком	ComboBox
5	Окно списка	ListBox
6	Флажок	CheckBox
7	Переключатель	OptionButton
9	Рамка	Frame
10	Кнопка	CommandButton
11	Набор вкладок	TabStrip
12	Набор страниц	MultiPage
13	Полоса прокрутки	ScrollBar
14	Счетчик	SpinButton
15	Рисунок	Image

Первая кнопка на панели – кнопка выбора объектов, которая используется для быстрого выделения нескольких объектов на форме. Нумерация элементов каждого типа ведется в пределах одной формы. Каждый элемент управления обладает собственным набором свойств и методов. Кроме того, существуют общие свойства, т.е. свойства, которые имеют все элементы управления. Наряду со свойствами и методами существует еще одно понятие, связанное с объектом – *событие*.

Событие представляет действие, распознаваемое объектом, для которого можно запрограммировать отклик (например, реакция системы на нажатие клавиши). Процедура, выполняемая при возникновении какого-либо события, называется *процедурой обработки события*.

Создание пользовательского приложения и будет заключаться, в основном, в написании кода процедур обработки событий для объектов, входящих в состав приложения.

Разработку приложения можно разбить на несколько основных этапов:

1. Формулировка задания. Уточнение структуры входных и выходных данных, вида их отображения на экране и в печатном виде.
2. Проектирование внешнего вида форм с общим описанием работы форм и их взаимодействия.

3. Описание событий форм и элементов, расположенных на формах и рабочих листах. Описание событий, связанных с рабочей книгой и рабочими листами.
4. Программирование процедур обработки событий и их отладка (возможно, с использованием режима автоматической записи макросов и их «ручной» доработки). Проверка работоспособности в случае ввода некорректных данных.
5. Тестирование работоспособности приложения.

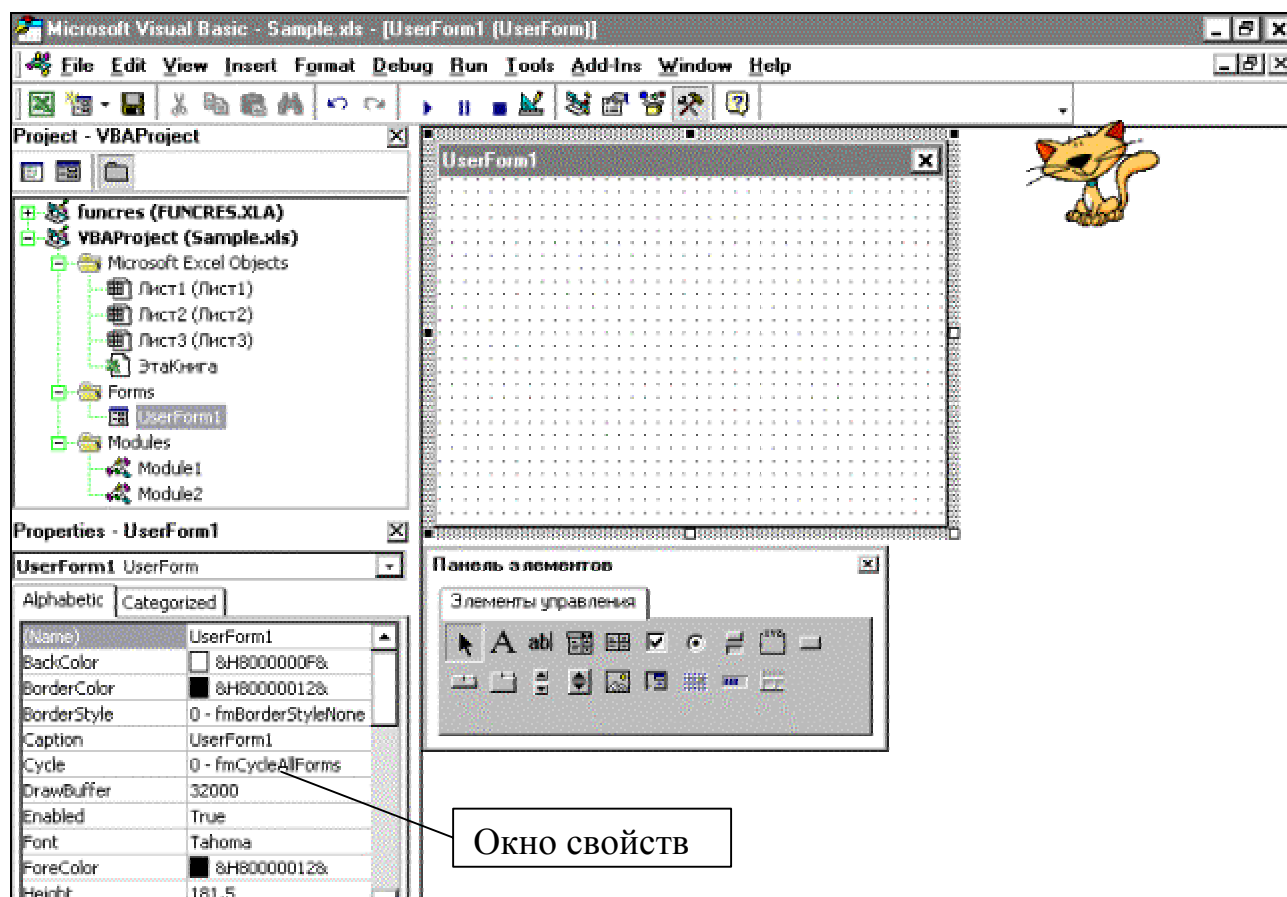


Рис. 18. Окно редактора VBA с пользовательской формой и панелью элементов

Свойства, методы, события

Свойства выделенного объекта отображаются в окне «Properties» редактора VBA (см. рис. 18). Если окно отсутствует на экране, нажмите кнопку F4. Свойства могут отображаться в окне в алфавитном порядке или сгруппированными по категориям. Для просмотра всего списка можно использовать полосу прокрутки.

Для установки свойства элемента необходимо выделить элемент, затем найти нужное свойство в списке и ввести или выбрать новое значение. Завершить действие нажатием Enter или переходом к другому свойству. Как видно из

рис. 18, свойств у выделенного объекта (в данном случае – Userform) достаточно много и нет необходимости пытаться сразу использовать все свойства. Рассмотрим основные *общие свойства* и *методы* элементов управления, чтобы в дальнейшем не дублировать их описание.

Общие свойства элементов управления

Name — Имя элемента управления

Имя обычно устанавливается по умолчанию при размещении элемента на форме, например, Listbox1, но если приложение содержит несколько форм, то трудно отличить первый список первой формы от первого списка второй формы. В этом случае лучше переименовать объекты. Для переименования объектов и переменных в среде Windows существует соглашение об именах, называемое венгерской нотацией. Данное соглашение состоит в следующем: рекомендуется начинать имя объекта с нескольких символов, одинаковых для всех объектов данного типа. За префиксом следует имя, отражающее суть данного объекта. Рекомендуемые префиксы имен приведены в таблице 17.

Таблица 17

Элемент	Префикс	Пример имени
TextBox	txt	txtFamily
Label	lbl	lblSaldo
Commandbutton	cmd	cmdWrite
Listbox	lst	lstWork
Combobox	cbo	cboGroup
ScrollBar	scr	scrKalory
Spinbutton	spn	spnWes
Optionbutton	opt	optDochod
CheckBox	chk	chkLgota
Image	img	imgFoto
MultiPage	mlp	mlpParam
Userform	frm	frmBudget

Enabled — Включить/отключить доступ к объекту.

Принимает значение True, если элемент доступен и False в противном случае. Если доступ отключен, элемент отображается серым цветом.

Visible — Показать/спрятать изображение объекта.

Принимает значение `True`, если элемент отображается на экране, `False` в противном случае. Это свойство позволяет управлять набором видимых объектов в зависимости от некоторых условий, например, чтобы не перегружать форму большим количеством элементов.

Height и Width – Устанавливают геометрические размеры объекта (высоту и ширину).

ControlTipText – Устанавливает текст в окне всплывающей подсказки, связанной с элементом управления.

Общие методы и события

Основные общие методы:

Add – Добавить объект во время выполнения программы.

SetFocus – Установить фокус на элементе управления. Достаточно часто применяется в процедурах обработки ошибочных ситуаций.

Наиболее часто употребляемые события связаны с возможными действиями пользователя.

Click – Событие происходит, когда пользователь выбирает элемент управления одинарным щелчком мыши.

DbClick – Событие происходит, когда пользователь выбирает элемент управления двойным щелчком мыши.

KeyPress – Событие происходит, когда пользователь нажимает любую клавишу на клавиатуре, кроме функциональных клавиш управления курсором.

Change – Событие происходит при изменении значения элемента управления.

GotFocus и LostFocus – Событие происходит, когда элемент управления получает или теряет фокус.

Краткое описание элементов управления

Надпись

Надпись (`Label`) позволяет размещать на форме подписи к элементам управления, а также дополнительную справочную информацию, которую пользователь не должен изменять.

Свойства:

- Caption** – Содержит текст надписи. Свойство доступно для чтения и записи.
- Autosize** – Если значение свойства установить True, то при изменении текста размер надписи будет изменяться по вертикали, т.е. будет изменяться количество строк для отображения текста.
- Font** – Устанавливает шрифт отображения текста надписи.

При использовании надписи в качестве поясняющего текста, свойство **Caption** изменяют сразу после размещения надписи на форме.

События для надписи обычно не обрабатываются, хотя, конечно, это возможно.

Примеры:

Пусть на **UserForm1** расположен элемент **Label1**.

1. Отобразить в надписи текущую системную дату:

```
UserForm1.Label1.Caption = Str(Date)
```

Функция **Str(аргумент)** преобразует аргумент к символьному типу.

2. Отобразить в надписи текстовую фразу:

```
UserForm1.Label1.Caption = «Скоро Новый год»
```

Кнопка

Кнопка (**CommandButton**) используется, в основном, для инициирования выполнения некоторых действий, вызываемых нажатием кнопки. Данный элемент управления присутствует в любом окне **Windows**. Основные свойства:

- Caption** – Устанавливает текст, расположенный на кнопке. Допускается чтение и запись свойства.
- Picture** – Рисунок, расположенный на кнопке.

Основное событие для кнопки – это, конечно, событие **Click**. Чтобы написать процедуру обработки данного события, можно дважды щелкнуть на кнопке, расположенной на **UserForm**, откроется окно кода с шаблоном процедуры, содержащем строки заголовка и конца процедуры. Например, в виде:

```
Private Sub CommandButton1_Click()
```

```
End Sub
```

Поле ввода

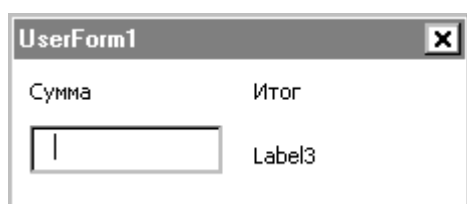
Свойства:

- Text** – Устанавливает или возвращает текст, содержащийся в элементе управления.
- Multiline** – Если установить значение True, то в поле будет отображаться многострочный текст.
- PasswordChar** – символ, отображаемый при вводе вместо действительного текста. Можно использовать для поля, в которое вводится пароль.
- LinkedCell** – Ячейка рабочего листа, в которую помещается текст каждый раз при его изменении.
- SelLenght, SelStart, SelText** – Перечисленные свойства характеризуют выделенный в поле фрагмент текста (длина, начало и сам фрагмент соответственно).

Текст, введенный в поле ввода, обычно проверяется в программе на соответствие указанному типу и при необходимости преобразуется в число или дату, а для текстовых значений можно, например, изменить регистр. Эти действия выполняются с помощью встроенных функций VBA.

Рассмотрим примеры использования поля ввода.

Пример1. Пусть создана форма, на которой расположены элементы: надпись, поле ввода. Написать процедуру, которая отображает в надписи значение, введенное в поле ввода с надбавкой 20%. Данную процедуру можно связать с событием **Change** поля ввода.



```
Private Sub TextBox1_Change()  
    x = Val(UserForm1.TextBox1.Text)  
    UserForm1.Label3 = Str(x + x * 0.2)  
End Sub
```

Рис. 19. Вид формы и текст процедуры

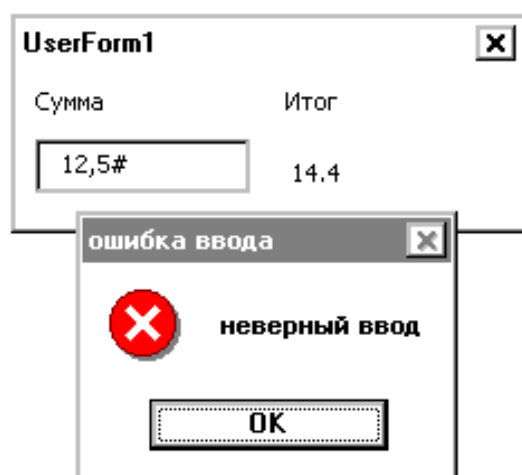
Обратите внимание на заголовок процедуры. Он состоит из имени элемента и имени события, которое обрабатывает данная процедура.

Чтобы проверить работоспособность процедуры, установите курсор внутри нее и нажмите F5. На экране отобразится созданная форма. Щелкните в поле ввода и введите различные значения. При вводе каждого следующего сим-

вола будет изменяться вычисленное значение. Событие **Change** происходит при вводе каждого символа.

Для завершения работы формы закройте ее обычным способом, как любое окно.

К сожалению, такой код процедуры предполагает, что данные в поле введены правильно. Рассмотрим возможный вариант контроля вводимых значений.



```
Private Sub TextBox1_Change()  
    If IsNumeric(UserForm1.TextBox1.Text) Then  
        x = Val(UserForm1.TextBox1.Text)  
        UserForm1.Label3 = Str(x + x * 0.2)  
    Else  
        MsgBox "неверный ввод", vbCritical + _  
vbOKOnly, "ошибка ввода"  
    End If  
End Sub
```

Рис. 20. Контроль ввода данных

Пример2. Данные, введенные в поле, можно записать в ячейку рабочего листа. Это реализуется обычным оператором присваивания. Например, запишем данные из полей «Фамилия» и «Имя» в ячейки B2 и C2 листа Список. Данная процедура может выполняться по щелчку на кнопке «Запись», т.е. обрабатывается событие **Click** командной кнопки.

```
Private Sub CommandButton1_Click()  
    Sheets("Список").Cells(2, 2).Value = UserForm2.TextBox1.Text  
    Sheets("Список").Cells(2, 3).Value = UserForm2.TextBox2.Text  
End Sub
```

В данном примере при создании формы у элементов надпись и кнопка было изменено свойство **Caption**.

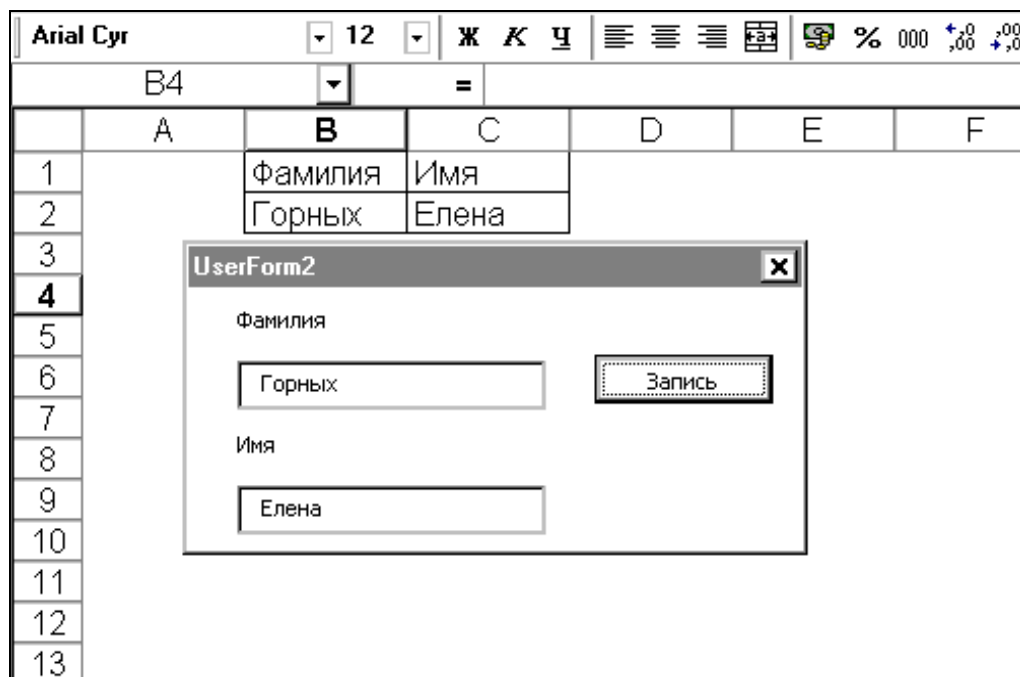


Рис. 21. Фрагмент экрана с изображением формы и результата

Список

Элемент управления список (ListBox) предназначен для хранения списка значений. Список часто встречается в стандартных диалоговых окнах (например, **Формат – Ячейки**, список числовых форматов). Пользователь имеет возможность выбирать из списка одно или несколько значений. В ряде случаев использование списка позволяет ограничить ввод данных с клавиатуры, избегая, таким образом, ошибочных ситуаций и упрощая процесс работы с приложением.

Наиболее употребительные свойства и методы списка предназначены для заполнения списка, определения значения выбранного элемента, его порядкового номера добавления и удаления элемента списка.

ListIndex – Номер выделенного элемента списка. Нумерация элементов списка начинается с нуля.

ListCount – Количество элементов списка.

RowSource – Диапазон, содержащий значения списка.

Value – Значение выбранного элемента списка.

MatchEntry – Устанавливает тип перемещения по элементам списка при вводе значения с клавиатуры. Возможные значения: режим контекстного перехода отключен, переход по первому набранному символу, по

полному набранному значению.

Варианты именуются специальными константами.

ColumnCount – Число столбцов в списке.

ColumnHeads – Установить/отменить режим отображения заголовков. При использовании диапазона для заполнения списка в качестве заголовков используются данные из строки над ним.

BoundColumn – Устанавливает номер столбца, из которого возвращается значение. По умолчанию равно 1. Если установить значение 0, то будет действовать, как **ListIndex**.

List – Массив значений, составляющих список.

Основные методы:

Clear – Очистить список, т.е. удалить все элементы.

AddItem – Добавить элемент в список.

RemoveItem – Удалить из списка элемент с указанным номером.

Поле со списком

Элемент поле со списком (**ComboBox**) аналогичен элементу список, но он сочетает в себе возможности списка и поля, т.к. позволяет выбирать и вводить значения. По-другому этот элемент иногда называют «одноточный список», т.к. он отображается в виде одной строки с кнопкой и позволяет выбрать только одно из множества значений. Свойства и методы в целом совпадают с рассмотренными ранее для списка. Кроме того, он имеет ряд уникальных свойств.

ListRows – Число строк раскрывающегося списка, т.е. количество элементов, которые отображаются после щелчка на кнопке.

Если элементов больше, чем указано, то автоматически добавится полоса прокрутки.

MatchRequired – Если установить значение **True**, то отключается возможность ввода и пользователь может только выбирать значения из списка.

MatchFound – принимает значение **True**, если среди элементов списка есть значение, совпадающее с введенным в поле ввода и **False** в противном случае.

Заполнение списка

Существует два способа заполнения списков:

- используя список на рабочем листе (установка свойства RowSource в окне свойств или программно);
- используя массив значений и методы AddItem и RemoveItem.

Основное требование состоит в том, что нельзя использовать оба способа одновременно для одного списка.

При записи ссылки на диапазон рабочего листа используются те же правила, что и при записи формул. Ссылка на диапазон при установке свойства RowSource может выглядеть, например, так: Данные!A1:A12. В качестве ссылки можно использовать имя диапазона. При изменении значения свойства в процедуре ссылка на диапазон записывается в двойных кавычках.

Примеры заполнения списка.

1. Установка свойства в окне свойств.

- a) Выделить список на форме.
- b) Найти строку свойства RowSource в окне свойств.
- c) Ввести ссылку на диапазон
- d) нажать Enter.

2. Программная установка свойства RowSource.

- a) для одноколоночного списка
`ListBox1.RowSource = "Данные!A1:A12"`
- b) для двухколоночного списка
`ListBox1.RowSource = "Данные!A2:B12"`

3. Поэлементное заполнение для одноколоночного списка.

```
ListBox1.AddItem "Школьник"  
ListBox1.AddItem "Учащийся"  
ListBox1.AddItem "Студент"
```

Для того, чтобы избежать многократного повторения имени одного объекта при последовательном выполнении нескольких операций, удобно использовать инструкцию With...End With, которая записывается следующим образом:

```
With объект  
    операторы  
End With
```

Можно сказать, что мы «выносим за скобки» имя объекта. Оператор, выполняющий некоторое действие с объектом, в данном случае должен начинаться с точки. Таким образом, пример, приведенный выше, можно переписать в виде:

With Listbox1

.AddItem "Школьник"

.AddItem "Учащийся"

.AddItem "Студент"

End With

4. Заполнение двухколоночного списка с использованием массива

a) With UserForm2.ListBox1

.ColumnCount = 2

.AddItem "Иванов"

.List(0, 1) = "Доцент"

.AddItem "Петров"

.List(1, 1) = "Профессор"

End With

b) Dim F(1, 1) As String

F(0, 0) = "Иванов"

F(0, 1) = "Доцент"

F(1, 0) = "Петров"

F(1, 1) = "Профессор"

ListBox1.ColumnCount = 2

ListBox1.List = F

Результат выполнения и в случае а), и в случае б), будет одинаковый, мы получим список, как на рис. 22.

Рис. 22. Заполненные списки

Для элемента поле со списком (ComboBox) действия по заполнению списка значениями записываются аналогично.

5. Удаление элементов из списка.

а) При использовании диапазона рабочего листа в качестве источника элементов списка добавление и удаление элементов списка выполняется обычными операциями работы со строками и ячейками. В данном случае удобно использовать именованные диапазоны, т.к. при добавлении и удалении ячеек внутри диапазона имя автоматически переопределяется. После изменения списка его можно отсортировать, чтобы восстановить нарушенный порядок записей. Если добавление должно осуществляться обязательно в конец списка, то после изменения списка надо переопределить значение свойства `RowSource`.

б) При удалении элементов с помощью метода `RemoveItem` указывается, с какой записи и сколько элементов удаляется.

Переключатель и флажок

Переключатель (`OptionButton`) позволяет выбрать один из предложенных вариантов действий или условий. Для обеспечения независимой работы нескольких групп переключателей, их располагают, заключая в рамку, используя элемент рамка (`Frame`). Основное свойство для переключателя – `Value`. Оно принимает значение `True`, если переключатель установлен и `False` в противном случае.

Флажок (`CheckBox`) позволяет установить или отменить указанное действие или параметр. Установка одного флажка не исключает установки других флажков на форме. Если флажок установлен, свойство `Value` принимает значение `True` и `False` в противном случае.

Текст, который выводится рядом с переключателем или флажком, содержится в свойстве `Caption`.

Основные события, которые обрабатываются для этих элементов – `Click` и `Change`.

Примеры:

Установить значение процентной ставки в зависимости от категории посетителя тренажерного зала. Если «Студент» – скидка 50%, «Преподаватель» – скидка 30%. Значение процента скидки отобразить в надписи.

Добавим на форму, созданную в примере заполнения списка (см. рис. 22), два переключателя и две надписи. Для каждого переключателя создадим процедуру обработки события `Click`.

```

Private Sub OptionButton1_Click()
    UserForm2.Label4.Caption = "50"
End Sub
Private Sub OptionButton2_Click()
    UserForm2.Label4.Caption = "30"
End Sub

```

Рис. 23. Использование переключателя

В данном окне при изменении положения переключателя изменяется значение процента скидки.

Добавим на форму еще несколько элементов – переключатель для категории «Другие» (скидка 0%), три флажка для выбора видов предоставляемых услуг, поле ввода и надпись для него, а также надпись для отображения итоговой суммы. Поле ввода заблокируем, установив для свойства **Enabled** значение **False**. Изменим положение некоторых элементов, чтобы сделать форму более наглядной.

Пусть стоимость каждой услуги задана числовой константой. При выборе определенной услуги ее стоимость добавляется к значению поля ввода «Сумма». Соответственно, при отказе от услуги, ее стоимость вычитается. При изменении положения переключателя, кроме изменения процентной ставки, будет обновляться значение «К оплате».

Рис. 24. Переключатели и флажки

```
Private Sub CheckBox1_Click()
```

```
    If UserForm2.CheckBox1.Value = True Then
```

```
        UserForm2.TextBox3.Text = Val(UserForm2.TextBox3.Text) + 10
```

```
    Else
```

```
        UserForm2.TextBox3.Text = Val(UserForm2.TextBox3.Text) – 10
```

```
    End If
```

```
    sk = Str(Val(UserForm2.TextBox3.Text) * Val(UserForm2.Label4.Caption)) / 100
```

```
    UserForm2.Label7.Caption = Str(Val(UserForm2.TextBox3.Text) – sk)
```

```
End Sub
```

```
Private Sub CheckBox2_Click()
```

```
    If UserForm2.CheckBox2.Value = True Then
```

```
        UserForm2.TextBox3.Text = Val(UserForm2.TextBox3.Text) + 20
```

```
    Else
```

```
        UserForm2.TextBox3.Text = Val(UserForm2.TextBox3.Text) – 20
```

```
    End If
```

```
    sk = Str(Val(UserForm2.TextBox3.Text) * Val(UserForm2.Label4.Caption)) / 100
```

```
    UserForm2.Label7.Caption = Str(Val(UserForm2.TextBox3.Text) – sk)
```

```
End Sub
```

```

Private Sub CheckBox3_Click()
If UserForm2.CheckBox3.Value = True Then
    UserForm2.TextBox3.Text = Val(UserForm2.TextBox3.Text) + 30
Else
    UserForm2.TextBox3.Text = Val(UserForm2.TextBox3.Text) - 30
End If
sk = Str(Val(UserForm2.TextBox3.Text) * Val(UserForm2.Label4.Caption)) / 100
UserForm2.Label7.Caption = Str(Val(UserForm2.TextBox3.Text) – sk)
End Sub
-----
Private Sub OptionButton1_Click()
‘ Переключатель «Студент»
UserForm2.Label4.Caption = "50"
sk = Str(Val(UserForm2.TextBox3.Text) * Val(UserForm2.Label4.Caption)) / 100
UserForm2.Label7.Caption = Str(Val(UserForm2.TextBox3.Text) – sk)
End Sub
-----
Private Sub OptionButton2_Click()
‘ Переключатель «Преподаватель»
UserForm2.Label4.Caption = "30"
sk = Str(Val(UserForm2.TextBox3.Text) * Val(UserForm2.Label4.Caption)) / 100
UserForm2.Label7.Caption = Str(Val(UserForm2.TextBox3.Text) – sk)
End Sub
-----
Private Sub OptionButton3_Click()
‘ Переключатель «Другие»
UserForm2.Label4.Caption = "0"
sk = Str(Val(UserForm2.TextBox3.Text) * Val(UserForm2.Label4.Caption)) / 100
UserForm2.Label7.Caption = Str(Val(UserForm2.TextBox3.Text) – sk)
End Sub
-----

```

В тексте процедур используются функции VBA Val – преобразование строки знаков в число и STR – преобразование значения в строку знаков.

Процедуры обработки события Click для флажков аналогичны друг другу, разница лишь в числовом значении параметра (для упрощения кода в данном примере мы считаем, что значение стоимости вида услуги задано константой).

Счетчик и полоса прокрутки

Элементы управления счетчик (SpinButton) и полоса прокрутки (ScrollBar) по своим функциональным характеристикам аналогичны друг другу. Основные свойства:

Value – текущее значение (целые неотрицательные числа).

Min – минимальное значение.

Max – максимальное значение.

SmallChange – шаг изменения значения.

LargeChange – только для полосы прокрутки. Шаг постраничного перемещения (при щелчке на свободной области полосы прокрутки).

Основные события – **Change** и **Click**.

Для отображения значения, установленного полосой прокрутки или счетчиком, можно использовать поле ввода или надпись.

Пример.

Установить значение возраста (полных лет) с помощью счетчика. Значение изменяется в диапазоне от 0 до 100.

Разместим на форме надпись, текстовое поле, счетчик. Установим значения свойств счетчика в окне свойств редактора VBA. Чтобы запретить внесение изменений в поле ввода, установим для свойства **Enabled** значение **False**. Запишем для счетчика процедуру обработки события **Change**.



```
Private Sub SpinButton1_Change()  
    UserForm3.TextBox1.Text = UserForm3.SpinButton1.Value  
End Sub
```

Рис. 25. Счетчик и текст процедуры

В данной процедуре текущее значение счетчика записывается в поле ввода и каждый щелчок на какой-нибудь кнопке счетчика приводит к изменению его значения и, соответственно, к изменению значения, отображаемого в поле ввода.

Набор страниц

При проектировании пользовательской формы важно не загромождать ее большим количеством элементов управления, что, несомненно, потребует увеличения ее размеров. В этом случае форма становится тяжелой для восприятия и работы.

Выдержать необходимый баланс элементов поможет набор страниц. Все элементы управления формы можно разделить на смысловые группы и разместить их на разных страницах, имеющих соответствующие заголовки (для примера откройте окно **Файл – Параметры страницы** или **Формат – Абзац**).

Основные свойства:

- Value** – Возвращает номер активной страницы. Нумерация страниц ведется с нуля.
- MultiRow** – Если установить значение **True**, то ярлыки страниц не помещающиеся в одну строку, будут выводиться в несколько строк, если **False**, то появляется полоса прокрутки, позволяющая переходить от страницы к странице.
- Caption** – Текст, отображаемый на ярлычке страницы.

Переход по страницам осуществляется щелчком мыши на ярлычке страницы. При размещении элемента **MultiPage** на форме по умолчанию создаются две страницы. Для добавления или удаления страницы открыть контекстное меню страницы и выбрать соответствующий пункт (см. рис. 26).

На каждой странице можно разместить уникальный набор элементов управления. В этом состоит основное отличие набора страниц от набора вкладок (**TabStrip**). Наборы вкладок используются для размещения одинаковых структур элементов управления.

Пример элемента **MultiPage** приведен на рис. 27.

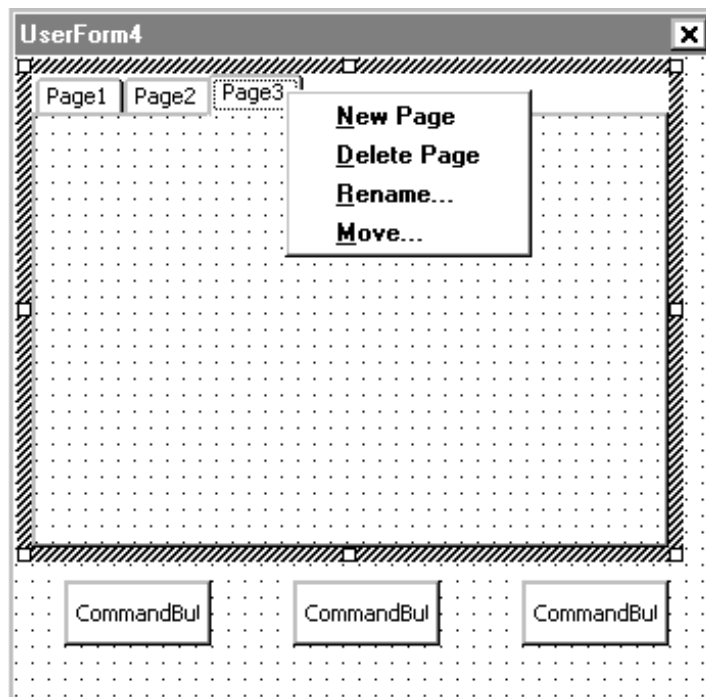


Рис. 26. Управление страницами

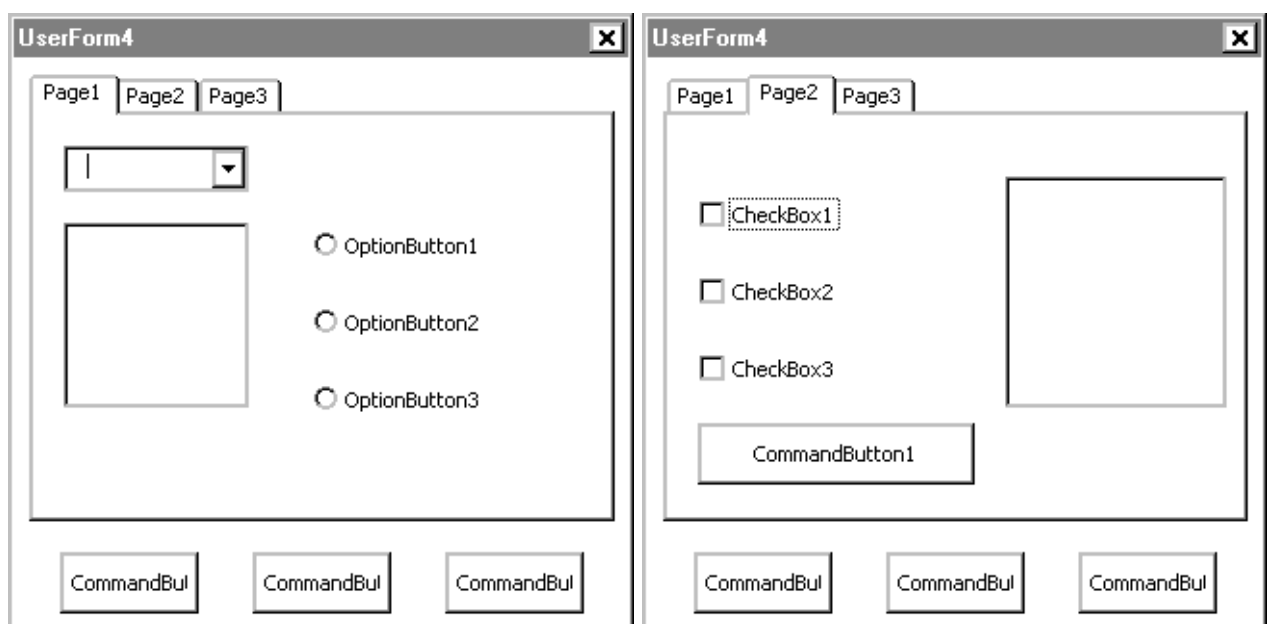


Рис. 27. Пример элемента MultiPage

Пользовательская форма

Пользовательская форма (UserForm) является базой для размещения элементов управления при создании диалогового окна. Наиболее часто используются свойства:

Caption – Текст, отображаемый в строке заголовка формы.

Height и Width – Высота и ширина формы в пунктах.

Основные методы формы:

Show – отобразить форму на экране.

Hide – закрыть форму.

Возможны два варианта отображения форм: в первом случае следующая форма отображается поверх предыдущей; во втором – в каждый момент времени на экране присутствует одна форма. Во втором случае перед отображением следующей формы необходимо закрыть предыдущую.

Основные события формы:

Initialize – происходит при первой загрузке формы.

Activate – происходит каждый раз при отображении формы на экране.

Terminate – происходит при закрытии формы.

Обработка событий Initialize и Activate обычно используется для установки начального состояния элементов управления на форме.

Пример.

Пусть имеется главное диалоговое окно, содержащее кнопки для вызова подчиненных окон. Тогда процедура обработки события Click для каждой кнопки будет содержать строку для отображения соответствующей формы на экране.

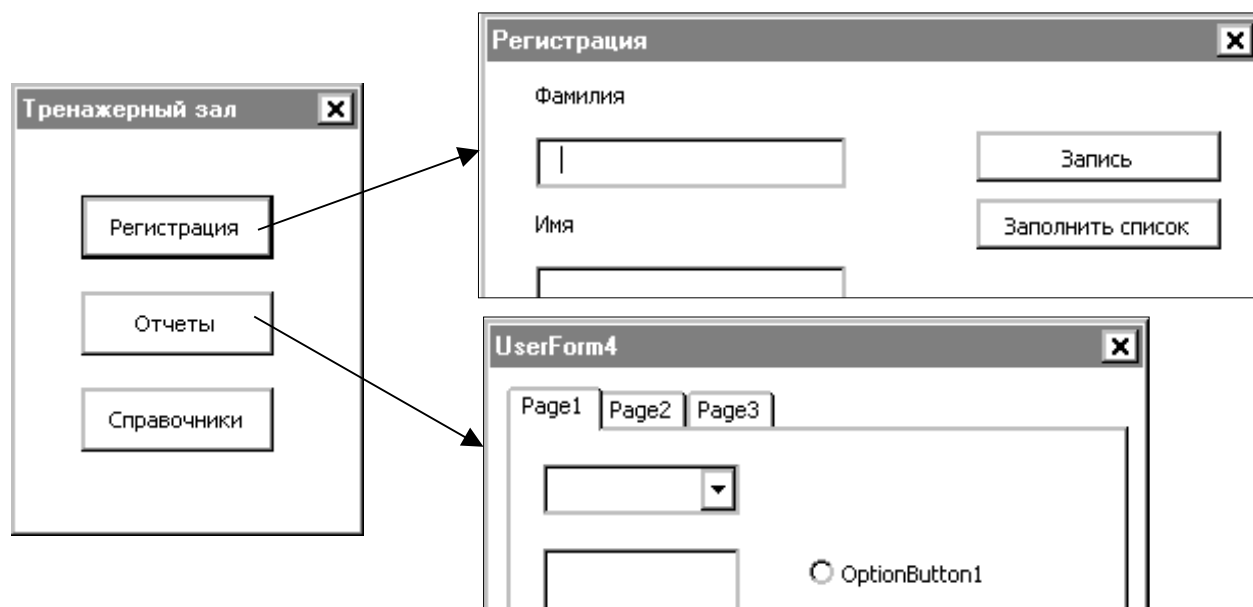


Рис. 28. Пример подчиненности форм

Процедура для кнопки «Отчеты», кроме вызова формы, содержит переход на нужную страницу. На рис. 28 приведены фрагменты подчиненных форм, рассмотренных в предыдущих примерах.

Тексты процедур:

```
Private Sub CommandButton1_Click()  
    UserForm2.Show  
End Sub  
Private Sub CommandButton2_Click()  
    UserForm4.MultiPage1.Value = 0  
    UserForm4.Show  
End Sub
```

Если необходимо отображать одну форму, то первая процедура, например, запишется в виде:

```
Private Sub CommandButton1_Click()  
    UserForm5.Hide  
    UserForm2.Show  
End Sub
```

В процессе работы приложения можно изменять размер отображаемой формы. Данный прием можно использовать, например, для отображения дополнительных элементов в зависимости от некоторых условий. Подобный прием используется, например, в окне «Вычисление поля сводной таблицы» для кнопки «Дополнительно>>». Размеры формы уточняются при ее проектировании, а затем изменяются программно. Продемонстрируем это на примере формы, приведенной на рис. 24. Добавим на форму кнопку «Услуги». При первоначальном отображении будет видна только верхняя часть формы. При щелчке на кнопке «Услуги» ее высота увеличится и отобразятся флажки и переключатели. Повторный щелчок на кнопке «Услуги» опять уменьшит высоту формы. Процедура обработки события Click для кнопки содержит анализ текущей высоты формы.

```
Private Sub CommandButton3_Click()  
    If UserForm2.Height = 303 Then  
        UserForm2.Height = 157  
    Else  
        UserForm2.Height = 303  
    End If  
End Sub
```

Обработка ошибок

Обработка ошибок является важным свойством профессиональных программ [8]. Для предотвращения некорректной работы программ существуют различные стратегии обработки ошибок. Достаточно условно можно выделить две стратегии:

- предупредительное программирование;
- перехват ошибок.

Чаще всего используется комбинация обеих стратегий.

С первой стратегией мы уже немного знакомы: проверка корректности ввода данных в текстовое поле.

Применяя предупредительную стратегию, программист пытается предупредить состояния, которые могут привести к ошибкам во время выполнения программы и избежать сбоя программы во время ее выполнения. Для этого можно использовать встроенные функции проверки свойств и значений и логические инструкции языка программирования.

Но не всегда возможно предугадать ситуацию. Например, при открытии файла возникнет ошибка, если файл отсутствует по указанному адресу.

Стратегия перехвата ошибок использует специальные команды VBA, которые предотвращают появление стандартных сообщений об ошибках и указывают на то, что вы намерены обработать ошибку самостоятельно. При этом указывается, какая часть кода будет выполняться при возникновении той или иной ошибки.

Иногда эта стратегия позволяет сократить код предупредительного программирования, который пришлось бы написать для обработки такой ситуации.

Для реализации таких действий используются следующие инструкции:

On Error Goto Label – используется как заголовок охраняемого блока. Устанавливается ловушка перехвата ошибок. Вместо генерирования ошибки управление передается метке, находящейся в той же процедуре (последняя установленная, наиболее локальная). **Label** – номер строки или метка.

On Error Resume Next – также используется как заголовок охраняемого блока, но с охраняемым блоком не связан обработчик ошибок. При возникновении ошибки управление передается оператору, следующему за ошибочным. Это оправдано в том случае, если далее стоит оператор, анализирующий возможные ошибки.

Если ошибка генерируется вне охраняемого блока, то выдается стандартное сообщение, предусмотренное системой.

On Error Goto 0 – данная инструкция удаляет любую ловушку.

Resume – инструкция является завершающей для обработчика ошибки и указывает, с какого места продолжить выполнение программы.

Она имеет четыре формы:

Resume – управление передается той инструкции, в которой произошла ошибка.

Resume Next – продолжить с оператора, следующего сразу после вызвавшего ошибку.

Resume Label – продолжить с указанной метки (номера строки). В тексте можно нумеровать не все строки.

Определение типа ошибки

Функция **ERR** возвращает целое число, соответствующее номеру ошибки (код ошибки). Коды ошибок можно посмотреть в Справочной системе. Например:

- 3 – Инструкция **Return** без **GoSub**.
- 11 – Деление на 0.
- 13 – Несоответствие типа.
- 55 – Файл уже открыт.

Для определение текста сообщения об ошибке используется функция **Error[(номер ошибки)]**. Она возвращает текст сообщения об ошибке. Если аргумент не указан, то текст сообщения для последней ошибки.

Определение места ошибки

Функция **ERL** возвращает номер последней строки, предшествующей сбойной. При использовании данной функции можно пронумеровать некоторые строки программы. Например:

```
Sub test2()  
1 On Error Resume Next  
2 z = 123 / 0  
3 MsgBox "ошибка номер " & Str(Err) & Error  
4 MsgBox Erl  
End Sub
```

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Запись макросов в автоматическом режиме

Для выполнения заданий можно использовать таблицы из учебного пособия [5] или файл с таблицами Tabl1cy.xls, находящийся в разделе «Практикум по Excel» WEB-хранилища кафедры «Информатика».

Задание 1

Создать в автоматическом режиме макрос фильтрации таблицы «Склад» по полю «Фирма» (расширенный фильтр с копированием результата на новое место), который реализует следующую последовательность действий:

1. Включить запись макроса
2. Очистить область результата фильтрации.
3. Выполнить расширенный фильтр.
4. Перейти к области результатов по ее имени.
5. Остановить запись макроса.

Прежде, чем включать запись макроса, выполните расширенный фильтр и убедитесь, что все получилось правильно. А уже затем реализовать п.п. 1-5

Познакомьтесь с операторами, записанными на листе «Модуль». Определите, какие действия они реализуют.

Вспомните возможные способы запуска макросов.

Создайте свою панель инструментов с оригинальной кнопкой, которой назначьте данный макрос.

Дополните макрос вопросом «Фильтровать?», который содержит кнопки ОК, ОТМЕНА и знак «?».

Проверьте ответ пользователя.

Дополните макрос вопросом «Введите название фирмы».

Если ответ верный, т.е. указана одна из трех фирм, содержащихся в таблице, то запишите значение в ячейку критерия фильтра.

Иначе выдать сообщение «Уточните название фирмы» и закончить макрос.

Проверить нажатие кнопки в окне ввода названия фирмы.

После сообщения «Уточните название фирмы» необходимо вернуться на ввод названия фирмы.

(Использовать Do While и флажок), причем в поле ввода должно отображаться введенное ранее значение.

Программа должна правильно воспринимать ответ пользователя, независимо от того, в каком регистре введено название— в верхнем, в нижнем, с большой буквы, т.е. правильными являются все значения:

ВЕСНА Весна весна ВеСнА и т.д.

Дополнить макрос операциями:

1. Вставить новый лист
2. Переименовать его в соответствии с названием фирмы
3. Обработать возможную ошибочную ситуацию (при наличии листа с таким именем)
4. Скопировать на новый лист отфильтрованную таблицу
5. Отменить перерисовку экрана во время выполнения фильтрации.

Задание 2

Расширенный фильтр. (таблица «Кадры»). Выбрать сотрудников отдела сбыта и бухгалтерии в возрасте старше 50 лет. Создать макрос, который выполняет фильтр для задания и копирует результат на лист «Отчет». Количество строк в таблице заранее неизвестно.

Задание 3

На листах Январь, Февраль, Март содержатся таблицы по потреблению электроэнергии для субъектов региона (Увельский р/н, Каслинский, Сосновский, Красноармейский.) Указаны: Норма потребления, общее потребленное количество, перерасход. Используя консолидацию получить отчет об общем потреблении и перерасходе электроэнергии.

Создать макрос, выполняющий консолидацию по заданию. Консолидированная таблица должна записываться на лист «Отчет»

Задание 4

Сводные таблицы. (таблица «Цех»). Составить отчет о среднем количестве больничных дней за каждый месяц. Создать макрос, который строит сводную таблицу на листе «Больничные».

Учесть, что таблица может иметь переменное количество строк.

Задание 5

Построить отчет в виде сводной таблицы, (таблица «Банк»), содержащий ежемесячные данные об общей сумме перечислений по 68, 69 счетам (сгруппировать по счетам). Вычислить отличие от предыдущего месяца.

Записать макрос, выполняющий задание на листе «Налоги».

Задание 6

Используя расширенный фильтр, (таблица «Магазин») выбрать товары, у которых срок годности истекает в ноябре. Записать макрос, который выполняет фильтр задания и копирует результат на лист «Уценка». Таблица содержит переменное количество строк.

Задание 7

Подведение итогов (таблица «Телефон»). Определить сумму оплаты и долга за каждый месяц.. Записать макрос, который копирует исходную таблицу на лист «Итоги» и выполняет подведение итогов по заданию. Таблица может содержать различное количество строк.

Задание 8

Подведение итогов. Составить отчет по каждой фирме о наименованиях и количестве полученного товара. Записать макрос, который копирует таблицу на лист «Фирма» и выполняет задание. Количество строк в таблице переменное.

Задание 9

Создать на трех листах ведомости реализации по магазинам Сатурн, Орбита, Зодиак за май и июнь. Таблицы содержат разное количество строк. Наименования повторяются. Используя консолидацию получить итоговую таблицу по суммарной реализации. Создать макрос, который строит консолидированную таблицу из задания 4 на листе «Отчет». Назначить макрос кнопке «Итог».

Редактирование макросов

1. Выполнить свой вариант задания по записи макросов в автоматическом режиме.
2. Создать личную панель инструментов.
3. Назначить макрос уникальной кнопке на созданной панели (изменить значок на кнопке). Всплывающая подсказка должна соответствовать назначению кнопки.
4. Дополнить макрос вопросом о подтверждении выполнения с возможностью отказа от действий.
5. Дополнить макрос вопросом о необходимости сохранения предыдущих результатов. В случае сохранения результатов новый результат записывать в свободной области (например, под старой таблицей)

Создание пользовательских приложений

Приложение "Семейный бюджет"

Создайте новую рабочую книгу.

Создайте на рабочем листе «Данные» список месяцев, список видов доходов и список видов расходов, список получателей.

Возможные значения для списка «Вид дохода»: стипендия, зарплата, пособие, взять в долг, % по вкладу, прочие.

Возможные значения для списка «Вид расхода»: оплата жилья, хоз. расходы, питание, развлечения, одежда, транспорт, возврат долга, прочие расходы.

Получатели: мама, папа, дочка, сын, все

Для любого из этих списков можно установить свои значения.

На листе «Бюджет» создайте шапку таблицы, начиная с ячейки A1.

№	Дата	Вид операции	Сумма дохода	Сумма расхода	Комментарий	Получатель
---	------	--------------	--------------	---------------	-------------	------------

Создайте три пользовательских формы в соответствии с рисунками. Последняя форма содержит восемь флажков и восемь надписей, содержание которых меняется в зависимости от параметров отчета, установленных в окне «Вид отчета». (Допускается замена последней формы двумя формами: отдельно по доходам и расходам).

Форма «Бюджет»

В зависимости от положения переключателя изменяется содержание списка «Вид операции». (обработать событие Click для переключателей).

Список «Получатель» не содержит элемент «Все».

Кнопка «Запись» реализует запись данных из элементов формы в таблицу на лист «Бюджет» (событие Click для кнопки «Запись»). Предусмотреть проверку корректности введенных значений. В случае ошибки выдать сообщение и установить курсор в ошибочное поле. Вычислить разность между суммой доходов и расходов по таблице на листе «Бюджет» и отобразить в надпись на форме. В зависимости от значения сальдо в соседнюю надпись выдается сообщение. После записи очистить поля «Сумма» и «Комментарий».

Рис. 29. Форма «Бюджет»

Кнопка «Удалить» реализует удаление последней записи таблицы. Предусмотреть подтверждающее сообщение с информацией о том, от какого числа и на какую сумму удаляется запись с возможностью отказа от удаления. Обновить сальдо при удалении.

Активизация формы. Устанавливаются начальные значения элементов управления. (дата, переключатель, списки, сальдо).

Кнопка «Закончить» сохраняет файл и закрывает его. Аналогично работает кнопка закрытия формы.

Кнопка «Отчет» отображает окно «Вид отчета»

Форма «Вид отчета»

Активизация формы. Происходит начальная установка значений переключателей и списков (значения, устанавливаемые по умолчанию).

Список «Получатель» содержит элемент «Все».

Переключатели «Общий», «Текущий», «За период» управляют доступностью соответствующих элементов. (Список «Получатели» доступен всегда).

Кнопка «Отчет». В зависимости от установленных значений параметров отчета формируется критерий расширенного фильтра для фильтрации исходной таблицы на листе «Бюджет» и выполняется фильтр с копированием результата на новое место.

В зависимости от переключателей «доход» и «расход» вычисляются суммы по каждому виду дохода или расхода и записываются в соответствующие метки третьей (или четвертой) формы.

Вид отчета

Период отчета

☒ Общий ☐ Текущий ☐ За период

Месяц: [dropdown] Получатель: [dropdown]

с [date] по [date]

☐ Доход ☐ Расход

Отчет Сводный отчет Заккрыть ?

Рис. 30. Форма «Вид отчета»

Дополнительно. Кнопка «Сводный отчет». Строится сводная таблица на листе «Сводный отчет» с учетом периода (общий, текущий или за период). Формы закрываются, чтобы дать пользователю возможность поработать с таблицей. Предусмотреть возможность вызова формы с листа или из панели инструментов. Если Excel был скрыт, то необходимо сначала показать Excel.

Кнопка «Заккрыть» закрывает форму и возвращает в форму «Бюджет», аналогично работает кнопка закрытия формы.

Кнопка «Справка» выдает информацию о работе с окном в виде формы или окна Помощника (объект Assistant не рассматривался в данном пособии, описание см. [1, 4]).

Форма «Отчет по доходам(расходам)»

Отчет по расходам

Вид операции	Сумма
<input type="checkbox"/> оплата жилья	881
<input type="checkbox"/> хоз.расходы	2015
<input type="checkbox"/> питание	2553
<input type="checkbox"/> развлечения	1130
<input type="checkbox"/> одежда	1660
<input type="checkbox"/> транспорт	165
<input checked="" type="checkbox"/> возврат долга	720
<input checked="" type="checkbox"/> прочие	1602

Заккрыть Подробнее Печать Справка

Рис. 31. Форма отчета по доходам (или расходам)

Если для отчетов используется одна форма, то ее содержание изменяется в зависимости от переключателей «Доход» и «Расход».

Заголовок формируется в соответствии с параметрами отчета.

Кнопка «Подробнее». В зависимости от установленных флажков формируется критерий выбора записей из таблицы, по которой получен отчет. Записи копируются на лист «Подробнее» и сортируются по виду операций и дате.

Предусмотреть возможности сохранения предыдущих отчетов или удаления старых данных. Таблицу показать пользователю. Предусмотреть возможность печати и возврата в форму (кнопки на листе). Лишние столбцы на листе «Подробнее» спрятать.

Кнопка «Печать». Печатает отчет в виде таблицы рабочего листа «Подробнее».

Кнопка «Справка». Выдает справку о работе с формой.

Общие требования к приложению

Все листы, кроме «Подробнее» (и «Сводный отчет»), спрятать.

Панели инструментов и строку формул спрятать.

Форма «Бюджет» загружается при открытии книги.

При закрытии книги восстанавливаются панели «Стандартная» и «Форматирование» и строка формул.

Запрос о сохранении изменений не должен появляться при закрытии приложения.

Вариант 1 «Ресторан в гостинице»

1. Используя средства Excel по созданию окон диалога и написание макросов на VB, создать приложение, автоматизирующее учет обслуживания клиентов в гостинице "Астория"

Исходные данные:

1. Ассортимент блюд с указанием группы (Закуски, первое, второе, напитки, выпечка, спиртное, десерт) и стоимости одной порции.
2. Список клиентов гостиницы с указанием номера проживания.

Приложение должно выполнять следующие функции:

1. Вести учет заказов от клиентов
2. Формировать счет на оплату заказа.
3. Формировать отчет о неоплаченных заказах.
4. Формировать счет клиенту с расшифровкой общей суммы.
5. Формировать отчет о суммах, полученных за период.
6. Формировать отчет о блюдах, пользующихся наибольшим спросом.
7. Изменять цену блюд и ассортимент.

Вариант 2 «Детский сад»

Используя средства Excel по созданию окон диалога и написание макросов на VB, создать приложение, автоматизирующее учет оплаты за детский сад в д/с «Солнечный зайчик».

Исходные данные:

Списки групп (4 Группы по 15 чел), стоимость одного дня пребывания.

№	Группа	Фамилия, имя	Кол-во дней	Пропущено	К оплате	Оплачено

Шаблон ведомости оплаты

Приложение должно выполнять следующие функции:

1. Учитывать количество дней к оплате и количество пропущенных дней. Реализовать удобную форму ввода пропущенных дней.
2. Формировать ведомость оплаты за месяц для каждой группы
3. Получать отчет о полученных суммах от каждой группы, количестве дней и пропущенных дней.
4. Формировать списки должников.
5. Вести учет за весь год
6. Учитывать изменения в списках групп

Вариант 3 «Библиотека»

Используя средства Excel по созданию окон диалога и написание макросов на VB, создать приложение, автоматизирующее учет книг в личной библиотеке.

Исходные данные:

1. Список книг с указанием количества экземпляров.
2. Список читателей библиотеки с указанием количества и названий книг на руках.

Приложение должно выполнять следующие функции:

3. Регистрировать выдачу книг в библиотеке. (Больше пяти— не выдавать!)
4. Регистрировать сдачу книг в библиотеку. (Показать, какие книги на руках).
5. Учитывать изменение остатка книг.
6. Учитывать срок выдачи и сдачи.
7. Формировать список должников на текущую дату.
8. Учитывать изменения в книжном фонде.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Биллиг В.А. Средства разработки VBA-программиста. Офисное программирование. – М.: Издательско-торговый дом «Русская редакция», 2001. – Т.1. – 480 с.
2. Биллиг В.А. VBA в Office 2000. Офисное программирование. – М.: Издательско-торговый дом «Русская редакция», 1999. – 480 с.
3. Гарнаев А. Использование Ms Excel и VBA в экономике и финансах. – СПб.: БХВ – Санкт-Петербург, 1999. – 336 с.
4. Гарнаев А. Самоучитель VBA. – СПб.: БХВ – Санкт-Петербург, 1999. – 512 с.
5. Горных Е.Н., Дудина Л.В. Практикум по работе с Excel: Учебное пособие. – Челябинск: Изд. ЮУрГУ, 2002. – 62 с.
6. Горных Е.Н., Дудина Л.В. Работа в MS Office: Учебное пособие. – Челябинск: Изд. ЮУрГУ, 2000. – 72 с.
7. Мэтью Харрис. Освой самостоятельно программирование для Microsoft[®] Excel 2000 за 21 день: Уч. пос./ Пер. с англ. – М. : Издательский дом «Вильямс», 2000. – 880 с.
8. Род Стивенс. Тестирование и отладка программ на Visual Basic/ Пер. с англ. – М. ДМК Пресс, 2001. – 384 с.
9. Э. Уэллс, С. Хешбаргер. Microsoft[®] Excel 97: разработка приложений/ Пер. с англ. – СПб.: БХВ – Санкт-Петербург, 1998. – 624 с.
10. Уильям Орвис. Visual Basic for Applications на примерах/ Пер. с англ. – М.: БИНОМ, 1995. – 512 с.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
ПРОСТЕЙШАЯ АВТОМАТИЗАЦИЯ	
МАКРОСЫ	4
РЕДАКТИРОВАНИЕ МАКРОСОВ	11
ОСНОВНЫЕ ОПЕРАТОРЫ ЯЗЫКА VBA	16
ФУНКЦИИ MSGBOX И INPUTBOX	23
ФОРМАТИРОВАНИЕ ЗНАЧЕНИЙ ДАННЫХ	28
ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ	31
ОБЪЕКТЫ, СВОЙСТВА, МЕТОДЫ	34
СЕМЕЙСТВА ОБЪЕКТОВ	36
ОБЪЕКТ APPLICATION	36
ОБЪЕКТ WORKBOOK	38
ОБЪЕКТ WORKSHEET И СЕМЕЙСТВО WORKSHEETS	39
СПОСОБЫ ДОСТУПА К ЯЧЕЙКАМ. ОБЪЕКТЫ RANGE И CELLS	40
СТАНДАРТНЫЕ ДИАЛОГОВЫЕ ОКНА	43
СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ПРИЛОЖЕНИЙ	43
СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ФОРМ	44
СВОЙСТВА, МЕТОДЫ, СОБЫТИЯ	46
КРАТКОЕ ОПИСАНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ	48
ОБРАБОТКА ОШИБОК	65
ПРАКТИЧЕСКИЕ ЗАДАНИЯ	
ЗАПИСЬ МАКРОСОВ В АВТОМАТИЧЕСКОМ РЕЖИМЕ	67
РЕДАКТИРОВАНИЕ МАКРОСОВ	69
СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ПРИЛОЖЕНИЙ	70
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	75