

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»



# МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ  
по дисциплине «Автоатизация офисных приложений»  
для студентов направления 09.03.03 «Прикладная  
информатика» (направленность (профиль) «Прикладная  
информатика в экономике»)

Ставрополь  
2020

## Содержание

Лабораторная работа №1. Типы данных, условные операторы и массивы VBA .....	3
1.2. Редактор VBA. Первое знакомство.....	8
1.3. Изменение порядка выполнения операторов.....	13
2.1. Свойства и методы объекта UserForm.....	21
2.2. Использование форм .....	28
Лабораторная работа №3. Массивы, процедуры, функции.....	37
3.1. Организация массивов .....	37
3.2. Работа с различными типами данных.....	42
3.3. Процедуры и функции VBA .....	48
Лабораторная работа № 4. Создание VBA-программ.....	58
4.1. Элемент управления ListBox .....	58
4.2. Элементы управления ComboBox, OptionButton и Frame .....	63
4.3. Элементы управления MultiPage, ScrollBar, SpinButton[2] .....	67
4.4. Объект DataObject[3].....	75
Лабораторная работа №5. Вывод текста в документ Word .....	82
5.1. Основные объекты Word .....	82
5.2. Форматирование документа .....	86
Лабораторная работа №6. Автоматизация стандартных документов .....	94
6.1. Встроенные диалоговые окна.....	94
Лабораторная работа №7. Работа с элементами activex .....	102
Лабораторная работа №7. Особенности использования VBA в Excel .....	107
7.1. Основные объекты VBA в Excel .....	107
7.2. Использование возможностей VBA при непосредственных расчетах.....	118
7.3. Финансовые функции.....	126
Лабораторная работа №8 Построение диаграмм средствами VBA.....	130
8.1. Построение гладких диаграмм.....	130
8.2. Построение круговых диаграмм и гистограмм .....	135
Лабораторная работа №9. Базы данных в Excel .....	139
9.1. Заполнение базы данных .....	139
9.2. Конструирование пользовательского интерфейса .....	143
Лабораторная работа № 10 Создание бланк заказа для обслуживания покупателей .....	148
Лабораторная работа №11 Модернизация бланка заказа .....	156
Лабораторная работа №12 Создание собственного головного меню .....	161

# Лабораторная работа №1. Типы данных, условные операторы и массивы VBA

## 1.1. Введение в VBA. Типы данных

### Введение в VBA

**Настоящий курс является начальным руководством по разработке программ, написанных на языке Visual Basic for Application (для приложений).**

VBA представляет собой набор средств программирования для создания собственных программ и подгонки имеющихся приложений под запросы пользователя.

Приложение – это полномасштабная программа, выполняющая конкретную практическую работу (например, текстовый процессор, электронные таблицы или приложение баз данных).

С помощью VBA можно изменять внешний вид или способ применения имеющихся средств приложения, а также добавлять свои, совершенно новые возможности.

В настоящее время VBA движется по направлению к тому, чтобы стать стандартом в индустрии создания программ. После освоения VBA вы сможете использовать этот язык в любом из приложений, поддерживающих VBA. Причем, зная VBA, вы автоматически изучаете язык Visual Basic.

Microsoft создала VBA и обеспечила поддержку VBA во всех главных приложениях Office: Word, Excel, Access и PowerPoint.

### Объектно-ориентированное программирование

Понимание объектов лежит в основе программирования в VBA, особенно когда дело касается создания пользовательских диалоговых окон и использования возможностей ведущего VBA-приложения.

### Определения

*Объектом* называется любая именованная сущность, имеющая:

- свойства, т. е. установки, которые можно проверить и изменить;
- методы, т. е. действия, которые может выполнить объект, когда программа попросит об этом;
- события, т. е. ситуации, в которых объект оказывается и на которые может ответить заранее определенными для таких ситуаций действиями.

*Коллекция* – это VBA-объект специального назначения. Коллекции предназначены для упрощения работы с набором объектов, когда этот набор объектов нужно использовать как одно целое. Как правило, все объекты в коллекции имеют один и тот же тип. Например, коллекция Pages состоит из объектов Page. Однако в VBA существует родовой объект Collection, предназначенный для хранения в нем объектов любых типов в любой комбинации.

*Формой* называют любое созданное в VBA пользовательское окно. Официально формы в VBA описываются в терминах объекта UserForm. Каждый объект UserForm принадлежит одновременно двум коллекциям объектов: VBA-проекту, в котором хранится форма, и коллекции UserForms, содержащей все формы, загружаемые программой.

*Свойства* – это характеристики объекта. Каждое свойство хранит информацию о некотором аспекте внешнего вида, поведения, содержимого объекта. Главной задачей свойства является описание некоторой характеристики объекта.

*Методы* – это именованные действия, которые объект может выполнить по команде. Ввиду того, что любой метод является неотъемлемой частью объекта, объект сам знает, что ему делать, когда вызывается метод. Таким образом, методы – не что иное, как процедуры, привязанные к конкретному объекту. Чтобы вызвать метод, необходимо напечатать имя объекта, точку, а затем имя метода.

*Событие* представляет собой нечто, случающееся с объектом, и то, на что объект может ответить заранее предусмотренным действием. К событиям можно отнести следующее:

- физические действия пользователя программы, например щелчок кнопкой мыши, перемещение курсора и т. д.;
- ситуации, в которые попадает объект в ходе выполнения программы.

Язык VBA является объектно-ориентированным. Это значит, что многие его команды имеют особенный формат. Типичная команда VBA имеет вид:

<Объект>.<Объект, входящий в первый объект>.<...>.<Тот объект, с которым нужно произвести действие>.<собственно действие>

Иными словами, каждая команда пишется как бы с «конца»: вначале определяется то, над чем надо произвести действие, – объект, а затем само действие – метод. Разделителями компонентов команды служат знаки «точка».

Например:

*Application.activDocument.PageSetup.Orientation=wdOrientLandscape*

Эта команда устанавливает альбомную ориентацию листа в документе.

### Обзор типов данных VBA

Тип данных – это термин, относящийся к определенным видам данных, которые VBA сохраняет и которыми может манипулировать. Любое определение типа задает:

- область возможных значений типа;
- структуру организации данных;
- операции, определенные над данными этого типа.

VBA разделяет обрабатываемые данные на числа, даты, строки, логические значения и объекты (табл. 1).

**Таблица 1 - Типы данных VBA**

Название типа	Размер в байтах	Описание и диапазон значения
Byte	1	Для хранения положительного числа от 0 до 255
Boolean	2	Для хранения логических значений; может содержать только значения True и False
Date	8	Для хранения комбинации информации о дате и времени. Диапазон может быть от 1 января 100 года до 31 декабря 9999 года. Диапазон времени от 00:00:00 до 23:59:59
Integer	2	Все целые числа от -32 768 до 32 767
Long	4	Все целые числа от -2 147 483 648 до 2 147 483 647
Single	4	Отрицательные числа от $-3,4 \times 10^{38}$ до $-1,4 \times 10^{-45}$ ; Положительные числа от $1,4 \times 10^{-45}$ до $3,4 \times 10^{38}$
String	1	Используется для хранения текста. Может содержать от 0 символов до приблизительно 2 миллиардов символов
Variant	16 байт + 1 байт/символ	Тип Variant может хранить любой другой тип данных. Диапазон для данных типа Variant зависит от фактически сохраняемых данных. В случае текста диапазон соответствует строковому типу; в случае чисел диапазон такой, как у типа Double
Double	8	Отрицательные числа от $-1,8 \times 10^{308}$ до $-4,9 \times 10^{-324}$ ; Положительные числа от $4,9 \times 10^{-324}$ до $1,8 \times 10^{308}$
Currency	4	Тип Currency используется для хранения чисел, когда точность крайне важна, что бывает при вычислениях с денежными единицами

Рассмотрим более подробно данные типы и выделим их яркие особенности.

VBA имеет шесть различных численных типов данных: Byte, Integer, Long, Single, Double и Currency. Численные типы данных используются для хранения (и манипулирования) чисел в различных форматах, в зависимости от конкретного типа. Численные типы предоставляют компактный и эффективный способ хранения чисел. Численный тип, заполняющий большую часть памяти (имеющий самый большой диапазон

возможных значений), занимает не более восьми байтов памяти для хранения чисел, которые могут иметь до 300 цифр.

Числа с плавающей точкой (*floating point numbers*) могут иметь любое число цифр до или после десятичной точки (в пределах границ конкретного типа данных) и получили свое название вследствие того факта, что десятичная точка «плавает» из одной позиции в другую, в зависимости от того, сохраняется в памяти большое или малое значение. Числа 24.09156, -1207.7, -0.00225 и 444.67779 – это числа с плавающей точкой. Числа с плавающей точкой иногда называют *также действительными (real)* числами. Данные типы с плавающей точкой используются всякий раз, когда требуется сохранить число, имеющее дробную часть. VBA имеет два различных типа данных с плавающей точкой: Single и Double.

Хотя числа одинарной и двойной точности имеют большие диапазоны, чем другие численные типы данных, у них имеются два небольших недостатка. Операции, выполняемые над числами с плавающей точкой, немного медленнее подобных операций над другими численными типами данных. Кроме того, числа, хранимые как типы данных с плавающей точкой, могут быть подвержены ошибкам округления. Если число с плавающей точкой очень большое или очень малое, VBA отображает его в экспоненциальном представлении.

VBA-тип *Currency* – это число с фиксированной точкой (*fixed-point number*), т. е. десятичная точка всегда находится в одном и том же месте – справа от десятичной точки всегда имеются четыре цифры. Тип *Currency* используется для хранения чисел, когда точность крайне важна, что бывает при «денежных» вычислениях.

Любые текстовые данные, сохраняемые в программе VBA, называются *строками (strings)*. Строки в VBA сохраняются с использованием типа данных *String*. Строки получили такое название, потому что текстовые данные обычно рассматриваются как строки символов. Строка может содержать текстовые символы любых типов: буквы алфавита, цифры, знаки пунктуации или различные символы. Строки в коде VBA всегда заключаются в двойные кавычки ("").

Существуют две категории строк: строки переменной длины, размер которых растет или уменьшается, и строки фиксированной длины, размер которых всегда остается одним и тем же. Все строки в VBA являются строками переменной длины, если только в программе не задается фиксированная длина.

Типы *String* играют важную роль во многих программах VBA. Большинство данных ввода пользователей (в диалоговых окнах, ячейках рабочих листов) – это строковые данные. Кроме того, поскольку на экране можно отображать только текст, все другие типы данных должны быть преобразованы в строковые данные перед тем, как они будут выведены на экран. Многие встроенные процедуры VBA используют строковые данные во всех или в некоторых своих аргументах.

VBA использует тип *Date* для хранения даты и времени. VBA-тип *Date* является типом *последовательных дат (serial Dates)*. (Последовательные даты сохраняют дату как число дней от заданной стартовой даты.) Базовой датой для VBA-типа *Date* является 30 декабря 1899. VBA использует отрицательные числа для представления дат ранее 12/30/1899 и положительные – для дат после 12/30/1899. Число 0 представляет саму дату 12/30/1899. По этой схеме 1 января 1900 записывается числом 2 (01/01/1900 – это 2 дня после 12/30/1899), однако число -2 является датой 12/28/1899 (два дня до 12/30/1899).

Даты можно вычитать одну из другой, добавлять к дате или вычитать числа для изменения ее значения. Например, если необходимо определить количество дней между двумя датами, просто необходимо вычесть более раннюю дату из более поздней даты. Поскольку это значения типа *Date*, VBA «знает», что целью вычисления является получение разности в днях между двумя этими датами. Аналогично, если необходимо определить дату через 60 дней после определенной даты, необходимо прибавить 60 к этой дате.

Во многих языках программирования, в том числе и в VBA-программе, рабочая программа должна «принять» решение, являются ли истинными различные условия. Для упрощения тестирования условий и обеспечения сохранения результатов такого тестирования в VBA имеется логический тип данных. Логические значения True и False называют *булевыми* (Boolean) значениями. Логический тип данных VBA называют также типом *Boolean*.

Тип данных *Variant* – это особый тип данных, который может сохранять любые типы, приведенные в табл. 1, за исключением типа *Object*. VBA использует тип Variant для всех переменных, если в теле программы не объявлялся явно тип этих переменных.

Данные типа Variant принимают характеристики определенного типа, который они сохраняют в этот момент. Например, если данные типа Variant содержат строковые данные, Variant принимает характеристики типа String. Если данные типа Variant содержат численные данные, Variant принимает характеристики какого-либо численного типа, обычно Double, хотя типы Variant могут также иметь характеристики типов Integer, Long, Single или Currency.

Несмотря на то что типы Variant удобны и избавляют от некоторой части работы при написании процедур, они требуют большего объема памяти, чем любой другой тип данных, за исключением больших строк. Кроме того, математические операции и операции сравнения над данными типа Variant выполняются медленнее, чем подобные операции над данными любого другого типа.

## Переменные

*Переменная* – это имя, которое разработчик программы дает области компьютерной памяти, используемой для хранения данных какого-либо типа. Переменная представляет числа, текстовые данные или другую информацию, которая точно не известна во время написания оператора, но будет в наличии и доступна при выполнении этого оператора.

*Идентификатор* – это имя переменной. При выборе имени переменной необходимо соблюдать следующие правила:

- имя переменной должно начинаться с буквы алфавита;
- после первой буквы имя переменной может состоять из любой комбинации цифр, букв или символов подчеркивания;
- имена переменных не могут содержать символов, используемых для математических операций, а также знака точки и пробела;
- имя переменной не должно превышать 255 символов;
- имя переменной не должно дублировать определенные ключевые слова vba.

Самым простым способом создания переменной является использование ее в операторе VBA. VBA создает переменную и тут же резервирует ячейку памяти для данной переменной.

Сохранение значения данных в переменной называется присваиванием переменной. Присваивание выполняется с помощью оператора присваивания, представляемого знаком (=). Например, A = 145.

Создание переменной путем ее использования в операторе называется *неявным объявлением переменной*. Все переменные, которые VBA создает неявным объявлением переменной, имеют тип данных *Variant*.

VBA предоставляет возможность выполнять *явное объявление* переменных. Объявлять переменные явно лучше в начале программы, как это делается во всех языках программирования.

Явно объявить переменную можно как в начале блока, так и в том произвольном месте, где возникла необходимость использовать новую переменную. При объявлении переменной определяются ее тип и область видимости – область, где имя переменной видимо и, значит, возможен доступ к ее значению. Переменные можно объявлять на двух уровнях – уровне процедуры и уровне модуля.

Для объявления переменных используются операторы Dim, Public, Private и Static. Первый можно использовать на обоих уровнях, Public, Private – на уровне модуля, Static – только на уровне процедуры.

Объявление простых переменных имеет следующий синтаксис:

Dim <имя переменной1, имя переменной2,...> As <имя типа>

или

Dim <имя переменной1> As <имя типа1>, <имя переменной2> As <имя типа2>, <имя переменной3> As <имя типа3>, ...

*Примечание.* Все переменные, которые создаются просто ключевым словом Dim, являются переменными типа Variant. Если используется ключевое слово As, то объявляемая переменная называется *типизированной*.

Примеры объявлений типизированных констант:

*Dim ключ As single*

*Dim стоимость As currency*

*Dim дата\_рождения AS Date*

*Dim письмо As string*

Явное объявление переменных имеет следующие преимущества:

- ускоряется выполнение кода, так как vba создает все объявленные явно переменные в модуле или процедуре перед выполнением кода процедуры;
- скорость выполнения кода увеличивается на то количество времени, которое необходимо для анализа и создания неявно объявляемых переменных;
- уменьшается количество ошибок в результате неправильного написания имени переменной;
- код становится легко читаемым и понятным, так как легко можно определить, какие переменные используются в этом модуле или процедуре.

*Примечание.* Имена переменных не «чувствительны» к состоянию регистра, т. е. переменные f и F в программе означают одно и то же.

## Константы

*Константа* – это значение в программе VBA, которое не меняется. Существует несколько типов констант.

*Именованные константы* – константы, имеющие заданное имя; это имя имеет конкретное неизменяемое значение.

В отличие от переменной, необходимо всегда явно объявлять именованные константы, используя ключевое слово Const.

Следует помещать объявления констант на модульном уровне, чтобы у них была наибольшая область действия.

*Литеральные константы* – это константы, записываемые непосредственно в код.

Правила написания литеральных констант (*String, Integer, Data, Boolean*):

- строковые константы должны быть заключены в двойные кавычки ("");
- пустая строковая константа (нулевая строка) обозначается двумя двойными кавычками, между которыми ничего нет ("");
- строковая константа должна вся находиться на одной и той же строке. Нельзя использовать символ продолжения строки для продолжения литеральной (строковой) константы на другой строке.

*Численные константы* могут содержать любой из численных типов VBA.

Правила написания численных констант:

- численные константы должны состоять только из числовых символов от 0 до 9;
- численная константа может начинаться со знака минус и содержать десятичную точку;
- можно использовать экспоненциальное представление для численных констант;
- никакие другие символы или знаки в численных константах не допускаются.

Примеры:

142

– 789.3

5.55E4

*Константы Date* необходимо помещать между знаками фунта (#). Независимо от того, в каком из форматов записывается литеральная константа Date, VBA переформатирует эту константу для соответствия одному из двух следующих форматов – в зависимости от того, содержит ли константа Date информацию о времени:

#2/5/02 9:17:00 PM#

#2/5/02#

*Константы Boolean* – существуют только две правильные константы типа Boolean: True и False.

*Типизированные константы* используются при явном задании типа константы. Объявление конкретного типа данных для константы может повысить точность вычислений. Для констант можно использовать типы данных Byte, Boolean, Integer, Long, Single, Double, Currency, Date, String.

Синтаксис:

Const имя\_константы As type = value, name As type = value,...

где type – имя любого из типов данных VBA;

value – значение, присваиваемое константе.

Пример:

Const Pi As Double = 3.14

*Внутренние константы* называются также предопределенными константами. Внутренняя константа – это именованная константа, которая была определена разработчиками VBA. Внутренние константы все начинаются с букв vb для указания того, что они определяются языком VBA (табл. 2).

**Таблица 2- Математические функции**

Функция	Возвращаемое значение
Abs(число)	Абсолютное значение числа
Atn(число)	Арктангенс числа
Cos(число)	Косинус числа
Exp(число)	Число e в степени, равной заданному числу
Fix(число)	Целая часть числа

Функция	Возвращаемое значение
Int(число)	Целая часть числа. Функции Int и fix по-разному действуют только на отрицательные числа: Int возвращает ближайшее меньшее целое, а Fix просто отбрасывает дробную часть числа
Log(число)	Натуральный логарифм числа, значение двойной точности
Rnd(число)	Случайное число, значение одинарной точности
Sgn(число)	1 — если число положительно, 0 — если равно нулю, -1 — если отрицательно
Sin(число)	Синус числа
Sqr(число)	Квадратный корень числа
Tan(число)	Тангенс числа

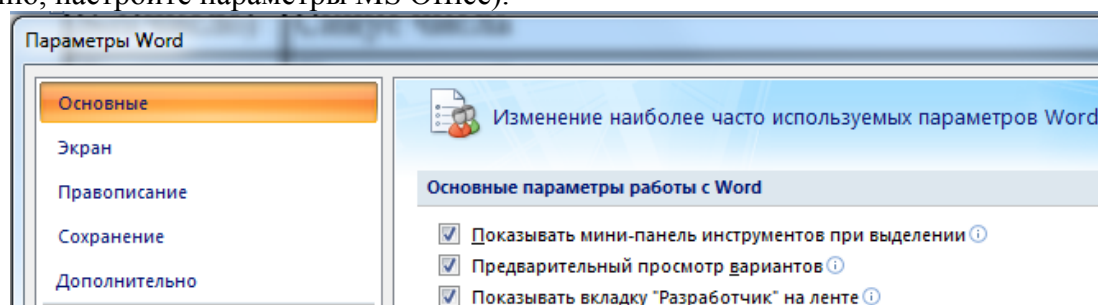
## 1.2. Редактор VBA. Первое знакомство

### Запуск редактора VBA

Как и любые среды программирования, редактор VBA необходимо сначала запустить. Для запуска можно использовать два способа:



- 1) активизировать любое приложение пакета MS Office (Word, Excel);
- 2) выполнить команду меню: Разработчик + Visual Basic (если данное меню не доступно, настройте параметры MS Office).



Или:

- 1) активизировать любое приложение пакета MS Office (Word, Excel);
- 2) нажать комбинацию клавиш Alt+F11.

И в том, и в другом случае откроется редактор VBA (рис. 1).

В левой части окна редактора появляется строение разрабатываемого проекта (аналог с Проводником). Необходимо обратить внимание на два главных объекта окна: Normal и Project (Операции).

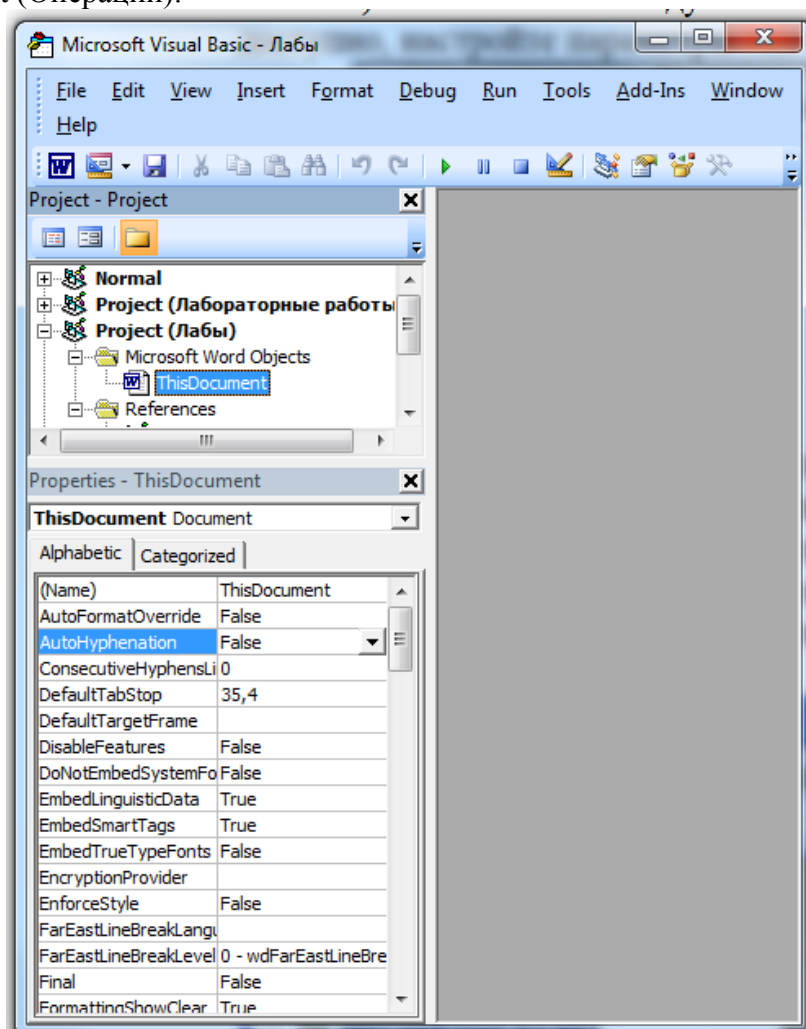


Рис. 1. Стартовое окно редактора VBA

*Примечание.* *Операции* – это имя сохраненного документа в приложении Word, т. е. в скобках будет указано имя сохраненного документа.

Объект Normal глобальный, т. е. при работе в редакторе VBA в данном объекте будут создаваться модули, формы и т. д., которые будут доступны всему приложению

Word. При каждом запуске Word содержимое объекта Normal становится доступным. Вывод: в данном объекте ничего не надо создавать!

Объект Project содержит рядом имя созданного документа, т. е. дается подсказка, в каком документе необходимо работать и где создаются модули, процедуры, приложения.

Создание простейших программ

*Пример 1.* Создать программу, которая работает с глобальными переменными, рассчитывая выражение:

$$d = \sqrt{\sin a^2 + \cos b + 3,14}.$$

Результат выдается в диалоговое окно MsgBox.

#### Технология выполнения

1. Активизируйте приложение Word, создайте (сохраните) новый документ под именем *Операции*.
2. Выйдите в редактор VBA (Alt+F11).
3. Правой кнопкой мыши выделите *Project(Document)*, где Document – название созданного документа, например *Операции*, и выберите команду *Insert + Module* (рис. 2). Если же имеется папка Modules, то добавьте в нее (рис. 6).

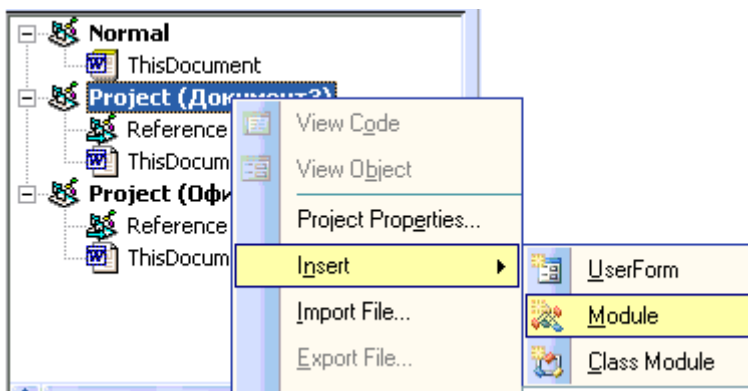


Рис. 2. Добавление нового модуля в папку *Project*

4. Дважды щелкните по классу *Module1* (рис. 3). В результате в правой части редактора VBA активизируется модуль (появится курсор), в котором можно прописывать все создаваемые программы, причем при вводе заготовки создаваемого модуля (слово Sub) и его имени (*list1* – от листинг, можно вписать любое имя) появляется окончание данного модуля *End Sub*.

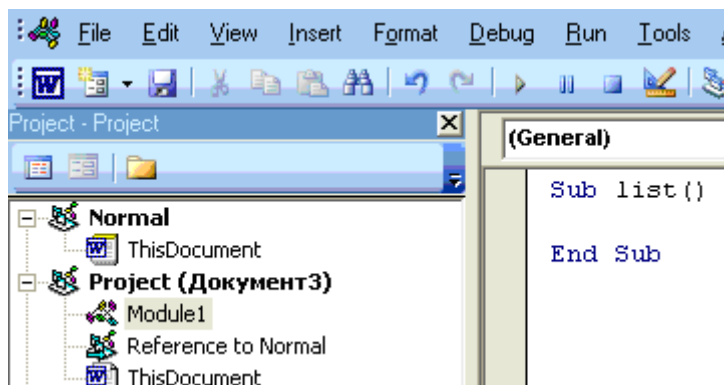


Рис. 3. Создание модуля *List1*

При объявлении глобальных переменных (согласно условию задачи) необходимо поставить курсор перед словом Sub, нажать Enter и вписать необходимые переменные, обращая внимание на раскрывающийся список (рис. 4).

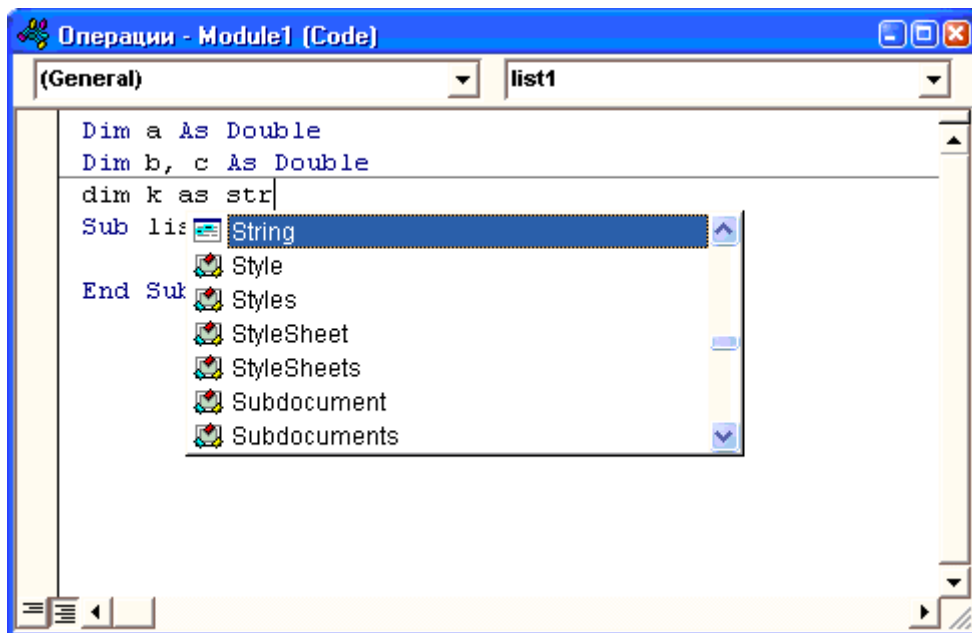


Рис. 4. Создание переменных 4. Напишите следующую программу.

```
Dim a As Double
Dim b, c, d As Double
Dim k As String
Const Pi As Double = 3.14
```

```
Sub list1()
a = 5
b = 25.6
c = Sin(a ^ 2) + Cos(b) + Pi
d = Sqr(c)
MsgBox (d)
End Sub
```

5. Запустите программу на выполнение, предварительно ее откомпилировав (проверив синтаксические ошибки). Для компиляции проекта выполните команду меню: Debug + Compile Project. Если ошибок нет, запустите приложение при помощи кнопки



Запуск панели инструментов или клавиши F5. Если все действия выполнены верно, то на экране появится диалоговое окно MsgBox с итоговым сообщением (рис. 5).

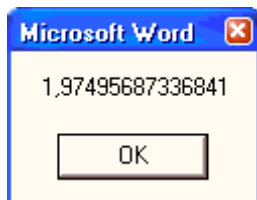


Рис. 5. Итоговый результат расчетов примера 1

*Пример 2.* Создать программу, которая производит сложение строковых переменных и результат выводит в диалоговое окно msgbox.

Технология выполнения

1. В том же документе *Операции* выделите правой кнопкой мыши папку *Modules* и выполните команду: *Insert + Module* (рис. 6).

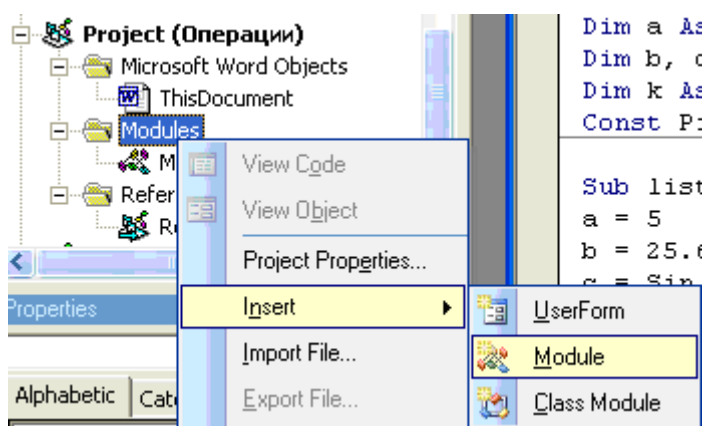


Рис. 6. Добавление нового модуля

2. В результате появится новый модуль, где можно создать новую программу (модуль). Пропишите следующий код.

```
Dim a, b, c As String
Sub list2 ()
a = "Привет!"
b = " Пока не сложно?"
c = a + b
MsgBox (c)
End Sub
```

В данном примере переменным а и b присваиваются строковые значения (в кавычках!), после чего происходит сложение строк. Результат показан на рис. 7.

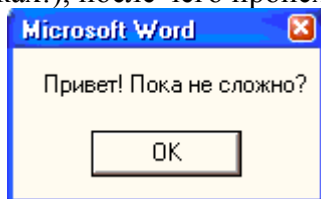


Рис. 7. Результат примера 2

3. Прокомпилируйте программу и запустите ее на выполнение.

*Примечание.* Если в одном документе находятся несколько модулей, то при запуске программ может появляться диалоговое окно выбора макроса (модуля), в котором необходимо выбрать макрос, выделить его и нажать кнопку «Run» (рис. 8).

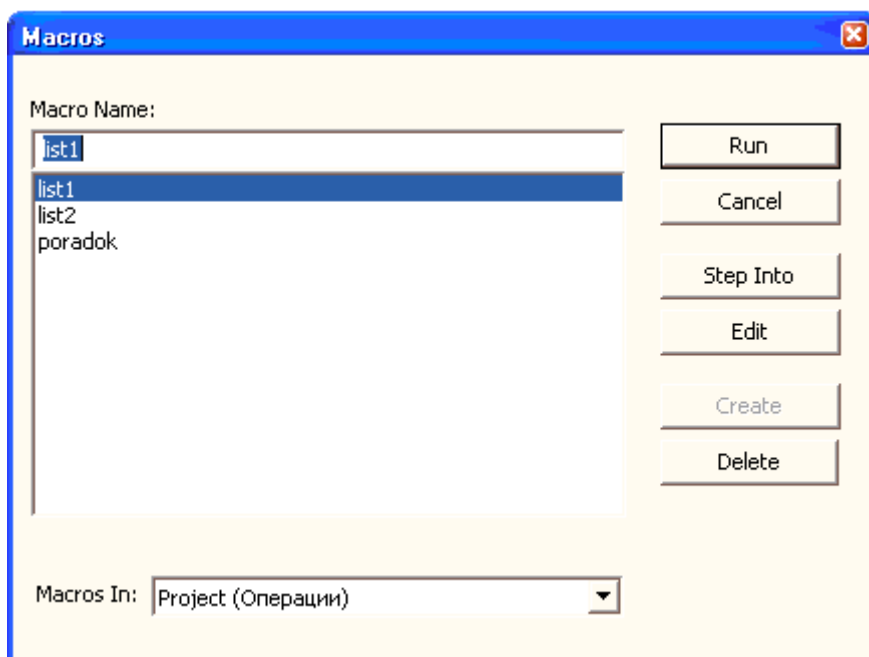


Рис. 8. Выбор макроса на выполнение

### 1.3. Изменение порядка выполнения операторов

Операторы и выражения

*Выражение* – это значение или группа значений, выражающая отдельное значение. Каждое выражение вычисляется до отдельного значения. Выражения состоят из одной или более следующих частей:

- константы (литеральные или именованные);
- переменные (любого типа данных);
- операторы;
- массивы;
- элементы массива;
- функции.

*Операторы* – используются для объединения, сравнения или других действий над определенными значениями в выражении. При использовании оператора в выражении элементы данных, над которыми этот оператор выполняет действие, называются операндами: большинству операторов требуются два операнда.

Выражение используется для выполнения вычислений и сравнения значений, для предоставления переменных в качестве аргументов различным функциям и процедурам VBA. Используются следующие типы выражений:

- выражение типа даты – вычисляется до значения типа date;
- численное выражение – вычисляется до любого числа;
- строковое выражение – имеет результатом значение типа string;
- логическое выражение – вычисляется до значения типа boolean.

Очень важно контролировать и знать тип выражения, потому что если выражения содержат несовместимые типы, VBA выдает ошибку времени исполнения – ошибку *несовпадения типов (type-mismatch)*. При обработке выражения, содержащего различные типы данных, VBA сначала пытается устранить любое различие типов, преобразуя значения в выражении в совместимые типы данных. Если устранить какие-либо различия преобразованием типов не удастся, отображается ошибка времени исполнения и процедура прекращает выполняться.

Рассмотрим основные арифметические операторы, используемые в VBA (табл. 3).

Таблица 3 - Арифметические операторы

Оператор	Описание
+, -, *	Сложение, вычитание, умножение
/	Деление
\	Целочисленное деление. При делении отбрасывает любую дробную часть так, чтобы результат был целым числом
Mod	Деление по модулю. Возвращает только остаток операции деления
^	Возведение в степень

Рассмотрим данные операторы более подробно.

#### *Сложение*

Оператор сложения (+) выполняет простое сложение. Оба операнда должны быть численными выражениями или строками, которые VBA может преобразовать в число. Оператор сложения можно также использовать для выполнения арифметических операций с данными типа Date.

#### *Вычитание*

Используется либо для вычитания одного числа из другого, либо для обозначения унарного минуса.

*Унарный минус (unary minus)* – это знак минус, который помещают перед числом (выражением) для указания того, что это отрицательное число (выражение). Поместить унарный минус перед числом или выражением означает то же, что умножить это число на минус 1.

#### *Умножение*

Оператор умножения (\*) перемножает два числа; результатом выражения умножения является произведение двух операндов. Оба операнда в выражении умножения должны быть численными выражениями или строками, которые VBA может преобразовать в число.

#### *Деление*

Оператор деления (/) называют оператором деления чисел с плавающей точкой или действительных, чтобы отличать от оператора целочисленного деления. Оператор деления с плавающей точкой выполняет обычное арифметическое деление двух операндов. В выражениях деления первый операнд делится на второй операнд; результатом деления является частное. Оба операнда в выражении деления с плавающей точкой должны быть численными выражениями или строками, которые VBA может преобразовать в числа.

#### *Целочисленное деление*

Выражения, использующие оператор целочисленного деления (\), всегда имеют результатом целое число без дробной части. Оба операнда в выражении целочисленного деления должны быть численными выражениями или строками, которые VBA может преобразовать в числа.

#### *Деление по модулю*

Деление по модулю (mod) дополняет целочисленное деление. В делении по модулю выражение возвращает только остаток операции деления как целое. Оба операнда в выражении деления по модулю должны быть численными выражениями или строками, которые можно преобразовать в число.

#### *Возведение в степень*

Оператор возведения в степень (^) возводит число в степень. Оба оператора в выражении возведения в степень должны быть численными выражениями или строками, которые VBA может преобразовать в числа. Операнд слева от оператора возведения в степень может быть отрицательным числом, только если операнд справа является целым. Если какой-либо операнд является равным Null, то результатом выражения возведения в степень также будет Null; иначе результат выражения будет иметь тип Double.

### Оператор Like

Оператор *Like* дает возможность выполнять особый тип операции сравнения строк, и его можно использовать только со строками.

Оператор *Like* тестирует строку для определения того, совпадает ли она с заданным шаблоном.

Синтаксис:

выражение1 *like* выражение2,

где *выражение1* – любое строковое выражение VBA;

*выражение2* – строковое выражение, специально созданное для задания шаблона, который оператор *Like* сравнивает с выражением1.

Результатом выражения *Like* является True, если первый операнд (*выражение1*) совпадает с шаблоном во втором операнде (*выражение2*); иначе результатом выражения будет False. Оба операнда в этом выражении должны быть строковыми выражениями, или VBA отобразит сообщение об ошибке несовпадения типов.

Шаблон, с которым должна сравниваться строка, задается с использованием различных специальных символов. В табл. 4 содержатся методы и символы для создания образцов совпадения для оператора *Like*.

Таблица 4 -Символы совпадения с образцом для оператора Like

Символ образца	Соответствие
#	Любая одиночная цифра от 0 до 9
*	Любое количество символов в любой комбинации или отсутствие символов
?	Любой одиночный символ
(list)	List — это список определенных символов. Совпадение с любым одиночным символом в списке
(! list)	List — это список определенных символов. Совпадение с любым одиночным символом, не имеющимся в списке

### Логические операторы

Чаще всего логические операторы используются для объединения результатов отдельных выражений сравнения, чтобы создать сложные критерии для принятия решений в процедуре, или для создания условий, при которых группа операторов должна повторяться (табл. 5).

Таблица 5 -Логические операторы

Оператор	Синтаксис	Описание
And	E1 And E2	Конъюнкция. Логическое выражение верно, если оба выражения верны (True)
Or	E1 Or E2	Дизъюнкция. Логическое выражение верно, если одно или оба выражения являются равными True; иначе выражение — False
Not	Not E1	Отрицание. Верно, если выражение имеет значение False; неверно (False), если выражение True
Xor	E1 Xor E2	Исключение. True, если первое выражение True или второе выражение является равным True; иначе — False
Eqv	E1 Eqv E2	Эквивалентность. True, если первое выражение имеет то же самое значение, что и второе; иначе — False
Imp	E1 Imp E2	Импликация. False, когда первое выражение является равным True и выражение 2 равно False; иначе — True

*Сложное (составное) выражение* – это любое выражение, образованное из двух или более выражений. Приоритеты выполнения операций при вычислении сложных выражений такие же, как и в любом языке программирования.

Изменение порядка выполнения операторов

Рассмотренные операторы выполняются в линейном порядке. При использовании VBA-операторов изменение порядка выполнения операторов определяется условием или набором условий, при которых VBA выполняет ту или иную ветвь кода процедуры.

*Оператор условного перехода* – это структура, которая выбирает ту или иную ветвь кода процедуры на основе некоторого предопределенного условия или группы условий.

*Оператор безусловного перехода* – это оператор, просто изменяющий последовательность выполнения кода процедуры независимо ни от какого конкретного условия. Условный переход используется гораздо чаще, чем безусловный.

Простейшими VBA-операторами изменения порядка выполнения кода являются операторы If ... Then и If ... Then ... Else.

Оператор If ... Then позволяет выбрать единственную альтернативную ветвь кода в процедуре или функции.

Синтаксис:

If условие Then оператор(ы),

где *условие* – любое логическое выражение;

*оператор(ы)* – один, несколько или ни одного оператора VBA.

Вторая форма синтаксиса оператора If ... Then называется блоком оператора if. В блоке оператора If ... Then условие и операторы записываются в отдельных строках, причем заканчивается данный оператор ключевыми словами End If.

Синтаксис:

If условие Then

оператор 1

оператор 2

....

Оператор n

End If

Выбор одной из двух различных ветвей операторов в зависимости от определенного условия обеспечивает оператор

If ... Then ... Else

и If ... Then ... ElseIf.

Синтаксис однострочного оператора If ... Then ... Else:

If условие Then оператор1 Else оператор2,

где *условие* – любое допустимое логическое выражение;

*операторы* – один или несколько операторов VBA, которые должны находиться в одной и той же строке.

Блок операторов If ... Then ... Else легче читать и понимать, и поскольку можно располагать операторы в разных строках внутри блока оператора If ... Then ... Else, он не имеет ограничения по размеру и числу операторов, которые можно помещать в альтернативные ветви.

Синтаксис:

If условие Then

Оператор1

Else

Оператор2

End If



VBA, как и многие языки программирования, имеет условный оператор перехода для использования в случаях, когда необходимо выбирать из большого количества различных ветвей кода: оператор Select Case. Данный оператор работает во многом так же, как и оператор If. Ключевые слова Select Case используются со многими операторами Case, где каждый оператор Case проверяет появление другого условия и выполняется только одна из ветвей Case. Ветвь Case может содержать один, несколько или ни одного оператора VBA.

Синтаксис:

Select Case выражение

Case условие\_1

Оператор\_1

Case условие\_2

Оператор\_2

.....

Case условие\_N

Оператор\_N

[Case Else

Оператор\_N+1

End Select,

где *выражение* – любое численное или строковое выражение;

*условие\_1, условие\_2, условие\_N* – (каждый) представляет список логических выражений, отделенных запятыми;

*оператор\_1, оператор\_2, оператор\_N, оператор\_N+1* – (каждый) представляет один, несколько или ни одного оператора.

В Select Case можно включать столько операторов Case условия, сколько необходимо.

*Примечание.* Написание нестроочных операторов в программах производится именно так (в столбец), либо редактор будет выдавать ошибку написания.

Оператор *безусловного перехода* всегда изменяет порядок выполнения операторов в процедуре или функции vba. При этом vba не проверяет никаких условий, а просто переходит к выполнению кода с другого места.

Оператор GoTo имеет следующий синтаксис:

GoTo *метка*

Метка – любое обозначение или номер строки в той же процедуре или функции, которая содержит оператор GoTo. При выполнении оператора GoTo VBA немедленно переходит к выполнению оператора в строке, определенной с помощью метки.

*Пример 3.* Создать программу, которая, используя инструкцию if ... then, выполняет следующие действия: если переменной a присваивается значение больше нуля, то находится сумма чисел a и b, если меньше нуля, то находится произведение. Результат выводится в стандартное диалоговое окно msgbox.

#### **Технология выполнения**

1. В документе *Операции* (пример 2) выделите правой кнопкой мыши папку *Modules* и выполните команду: *Insert + Module* (рис. 6).

2. В появившемся модуле пропишите программу и запустите на выполнение.

```

Dim a, b, c As Integer
Sub poradok()
a = -5
b = 25
If a > 0 Then
c = a + b
MsgBox (c)
End If
If a < 0 Then
c = a * b
MsgBox (c)
End If
End Sub

```

В данной программе переменной *a* присвоено значение меньше нуля, следовательно, должна выполняться нижняя инструкция If (рис. 9).

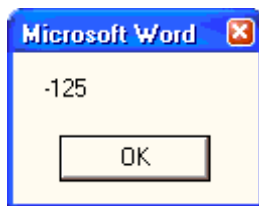


Рис. 9. Результат работы при  $a < 0$

3. Измените программу, поменяв значение *a* на положительное:

```

Dim a, b, c As Integer
Sub poradok()
a = 5
b = 25
If a > 0 Then
c = a + b
MsgBox (c)
End If
If a < 0 Then
c = a * b
MsgBox (c)
End If
End Sub

```

*Примечание.* Данную программу можно составить, используя полный блок инструкции If Then Else (рис. 10).

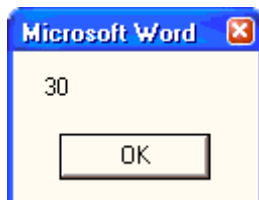


Рис. 10. Результат работы при  $a > 0$

```

Dim a, b, c As Integer
Sub poradok()
a = 5
b = 25
If a > 0 Then
c = a + b
MsgBox (c)
Else
c = a * b
MsgBox (c)
End If
End Sub

```

### Повторение действий: циклы

Процесс выполнения всех операторов, заключенных в структуру цикла, один раз называется *итерацией (iteration)* цикла. Некоторые структуры цикла организуются так, что они всегда выполняются заданное количество раз. Структуры цикла, всегда выполняющиеся заданное количество раз, называются циклами *с фиксированным числом итераций (fixed iteration)*. Другие типы структур цикла повторяются переменное количество раз в зависимости от некоторого набора условий. Поскольку количество раз повторений этих гибких структур цикла является неопределенным, такие циклы называются *неопределенными циклами (indefinite loops)*.

Существуют два основных способа создания неопределенного цикла. Можно построить цикл так, что VBA будет тестировать некоторое условие (детерминант цикла) *перед* выполнением цикла. Если условие для повторения цикла не равно True, VBA пропускает все операторы в цикле. Можно также построить цикл таким образом, что VBA будет тестировать условие детерминанта цикла *после* выполнения операторов в цикле.

Самой простой структурой цикла является фиксированный цикл. VBA предоставляет две различные структуры фиксированного цикла: For... Next и For Each ... Next. Обе структуры фиксированного цикла называются циклами For, потому что они всегда выполняются *для (for)* заданного количества раз.

#### Использование цикла For... Next

Цикл For...Next используется, когда необходимо повторить действие или ряд действий заданное количество раз, известное до начала выполнения цикла.

Цикл For...Next имеет следующий синтаксис:

For *a* = *Start* To *End* [Step *StepSize*]

операторы

Next [*a*],

где *a* – любая численная переменная VBA, обычно переменная типа Integer или Long;

*Start* – любое численное выражение, определяет начальное значение для переменной

*a*;

*End* – это также численное выражение, определяет конечное значение для переменной *a*.

По умолчанию VBA увеличивает переменную *a* на 1 каждый раз при выполнении операторов в цикле (считает количество циклов). Можно задавать другое значение (*StepSize*), на которое будет изменяться *a*, включая необязательное ключевое слово Step. При включении ключевого слова Step необходимо задавать значение для изменения переменной *a*.

*Операторы* представляют один, несколько или ни одного оператора VBA. Эти операторы составляют тело цикла For; VBA выполняет каждый из этих операторов каждый раз при выполнении цикла.

Ключевое слово Next сообщает VBA о том, что достигнут конец цикла; необязательная переменная *a* после ключевого слова next должна быть той же самой переменной *a*, которая была задана после ключевого слова For в начале структуры цикла.

Включайте необязательную переменную *a* после ключевого слова next для улучшения читабельности программного кода (особенно при использовании вложенных циклов for next) и для повышения скорости выполнения кода (иначе vba придется потратить время на определение того, какая переменная является правильной, для ее изменения после достижения ключевого слова next).

#### *Цикл For Each ... Next*

Второй цикл For, который имеется в VBA, – это цикл For Each ... Next. В отличие от цикла For...Next, цикл For Each ... Next не использует счетчик цикла. Циклы For Each ... Next выполняются столько раз, сколько имеется элементов в определенной группе, такой как коллекция объектов или массив. Другими словами, цикл For Each ... Next выполняется один раз для каждого элемента в группе.

Цикл For Each ... Next имеет следующий синтаксис:

For each *a* in группа

*операторы*

Next [*a*],

где *a* – это переменная, используемая для итерации по всем элементам в определенной группе;

*группа* – это объект коллекции или массив. Если *группа* – это объект коллекции, то *a* должна быть переменной типа variant, object или заданным объектным типом, таким как range, worksheet, document, paragraph и т. д.

Если *группа* – это массив, то *a* должна быть переменной типа variant;

*операторы* – один, несколько или ни одного оператора VBA, составляющих тело цикла.

*Примечание.* Примеры на использование циклов будут рассмотрены в части II.

## Лабораторная работа № 2. Объект UserForm

### 2.1. Свойства и методы объекта UserForm

#### Свойства объекта UserForm

Рассматривая в п. 1.2 и 1.3 примеры, решаемые в редакторе VBA, можно прийти к выводу, что такие простейшие задачи с выводом единственного результата в диалоговое окно не всегда будут удовлетворять потребностям пользователей. Одним из достоинств языка программирования VBA является то, что он относится к объектно-ориентированным языкам. Следовательно, в данную среду программирования уже заложены возможности создания форм и его элементов простым использованием без составления громоздкого программного кода.

Практически во всех приложениях Office используются пользовательские диалоговые окна. Диалоговые окна в VBA называются формами (объект UserForms). Каждому объекту UserForm присущи определенные свойства, методы и события, которые он наследует от класса объектов UserForms. Диалоговые окна (формы) и элементы управления составляют основу современного визуального интерфейса. Все элементы управления и технология работы с ними в основном стандартизованы и похожи для разных платформ и программных сред. Эти объекты помещены в специальную библиотеку MSForms.

Выделим основные моменты, которые следует иметь в виду при создании визуального интерфейса.

- Все загруженные диалоговые окна представляют коллекцию *UserForms* со стандартными методами и свойствами. Элемент коллекции – объект класса *UserForm* – задает отдельное окно.
- Для каждого типа элементов управления в библиотеке msforms имеется класс объектов, имя которого совпадает с именем элемента управления (его типа). Например, есть классы *SpinButton* и *TextBox*.
- Диалоговые окна создаются, как правило, не программно, а визуально. Вначале создается само окно, а затем оно наполняется элементами управления при помощи соответствующей панели элементов. Этот этап называется *этапом проектирования*, и его следует отличать от этапа *выполнения*, когда приложение выполняется и конечный пользователь взаимодействует с приложением, в частности через диалоговые окна и их элементы управления. Как только создается диалоговое окно и помещается в него тот или иной элемент управления, в этот же самый момент автоматически в программе появляется объект соответствующего класса, с которым можно работать, вызывая его методы и изменяя его свойства.

На этапе проектирования, используя окно свойств, можно задать большинство свойств как самого диалогового окна, так и всех элементов управления, помещенных в него, кроме этого, программно необходимо прописать все обработчики событий.

- Последний момент – отладка. Для ведения отладки нужно предварительно откомпилировать приложение и затем перейти в режим выполнения приложения.

Для того чтобы в разрабатываемое приложение можно было добавить форму, необходимо выполнить следующие действия:

- 1) запустить редактор VBA;
- 2) выделить правой кнопкой мыши объект Project, выполнить команду Insert + UserForm (рис. 11), после чего появляются новая форма и панель элементов Toolbox (рис. 12).

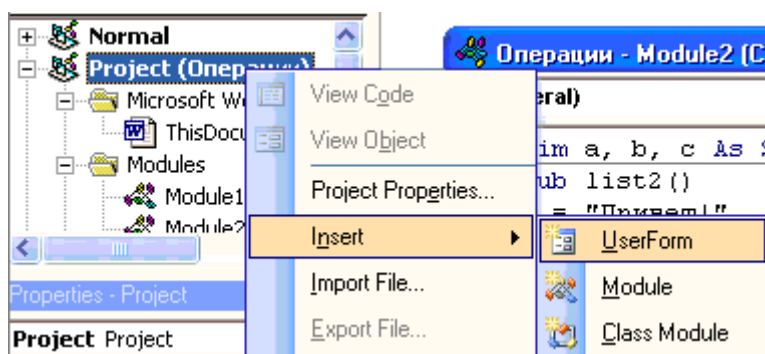


Рис. 11. Добавление формы

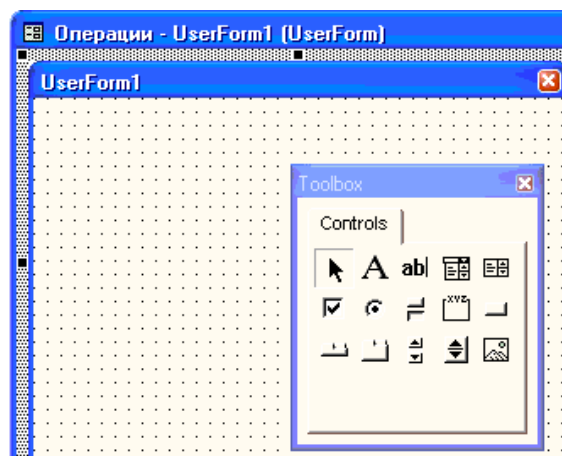


Рис. 12. Новая форма

Форма как объект имеет некоторые встроенные свойства, и их можно устанавливать или программным образом, или в Properties Window (окне свойств) редактора VBA (табл. 6).

Таблица 6- Наиболее часто используемые свойства объектов UserForm

Свойство	Описание
ActiveControl	Возвращает объектную ссылку на элемент управления, находящийся в фокусе в данный момент. Только для чтения
BackColor	Целое типа Long определяет цвет фона формы
Caption	Текст, выводимый в качестве заголовка формы
Controls	Возвращает коллекцию всех элементов управления формы
Cycle	Определяет, должно ли нажатие клавиши табуляции вызывать последовательный выбор всех элементов управления во всех группах и на каждой странице многостраничных элементов управления или только в пределах текущей группы или страницы. Может содержать одну из двух встроенных констант: fmCycleAllForms или fmCycleCurrentForm
Enabled	Содержит значение типа Boolean, указывающее, доступна ли форма. Если его значение равно False, ни один из элементов управления формы не доступен

Свойство	Описание
Font	Возвращает ссылку на объект Font, посредством которого можно выбрать параметры шрифта формы или элемента управления
ForeColor	То же самое, что и свойство BackColor, но устанавливает цвет, используемый для переднего плана (обычно это цвет текста) объекта формы

### Методы объекта UserForm

Всякий раз, создавая в проекте новый объект UserForm, одновременно создается новый подкласс объекта UserForm. Любые процедуры или функции, написанные в разделе General (общий) модуля класса, относящегося к форме, становятся дополнительными методами для отдельного подкласса объекта (табл. 7).

Таблица 7 -Наиболее часто используемые методы для объектов UserForm

Метод	Назначение
Copy	Копирует выделенный в элементе управления текст в буфер обмена Windows
Cut	Вырезает выделенный в элементе управления текст и помещает его в буфер обмена Windows
Hide	Скрывает UserForm, не выгружая ее из памяти, сохраняя значения элементов управления формы и всех переменных, объявленных в модуле класса формы
Paste	Вставляет содержимое буфера обмена Windows в текущий элемент управления
PrintForm	Выводит на используемый в Windows по умолчанию принтер изображение формы, включая все данные, введенные в элементы управления
Repaint	Перерисовывает форму, выведенную на экран. Используется этот метод, если необходимо перерисовать форму, не ожидая, когда она будет перерисована через обычный период времени
Show	Выводит форму на экран. Если форма еще не загружена в память, то данный метод сначала ее загружает. Синтаксис метода Show: <i>FormName.Show</i>

### События объекта UserForm

*Событие* – это что-то, что может произойти с диалоговым окном или элементом управления диалогового окна (табл. 8).

Событийные процедуры следует записывать в модуль класса, который является частью User Form. При этом такие процедуры должны иметь имена в виде

ObjectName\_EventName,

где *ObjectName* – имя формы или элемента управления, а *EventName* – имя события, с которым идет работа. Такой формат имени позволяет VBA сопоставлять заданному событию требуемую процедуру.

Таблица 8 -События объектов UserForm














Событие	Синтаксис заголовка процедуры	Описание
Activate	Private Sub <i>object_Activate()</i>	Иницируется всякий раз, когда окно формы становится активным. Используйте это событие для обновления содержимого диалоговых элементов управления, чтобы отразить любые изменения, которые произошли, пока окно формы было неактивным
Click	Private Sub <i>object_Click()</i>	Иницируется всякий раз, когда по форме (любой ее части, не занятой элементами управления) щелкают мышью
DblClick	Private Sub <i>object_DblClick()</i>	Иницируется всякий раз, когда по форме (любой ее части, не занятой элементами управления) дважды щелкают мышью
Deactivate	Private Sub <i>object_Deactivate()</i>	Иницируется всякий раз, когда форма перестает быть активной
Initialize	Private Sub <i>object_Initialize()</i>	Иницируется всякий раз, когда форма впервые загружается в память посредством выполнения оператора Load или с помощью метода Show. Используйте это событие для инициализации элементов управления формы при ее появлении на экране
Resize	Private Sub <i>object_Resize()</i>	Иницируется при изменении размеров формы
Событие	Синтаксис заголовка процедуры	Описание
Terminate	PrivateSub <i>object_Terminate()</i>	Иницируется всякий раз, когда форма выгружается из памяти. Используйте это событие для осуществления любых специальных служебных задач, которые необходимо выполнить прежде, чем переменные формы будут выгружены


### Элементы управления

Объект UserForm может содержать те же элементы управления, что и находящиеся в диалоговых окнах Word, Excel или других приложений Windows (табл. 9). *Элементы управления* – это элементы диалогового окна, позволяющие пользователю взаимодействовать с программой. Они включают в себя кнопки-переключатели, текстовые поля, линейки прокрутки, командные кнопки и т. д.

Таблица 9 - Стандартные элементы управления, включенные в VBA



Элемент управления	Назначение
Label  (надпись, метка)	Позволяет создавать заголовки элементов управления, которые не имеют собственных встроенных заголовков
TextBox  (текстовое поле)	Окно редактируемого текста свободной формы для ввода данных. Может быть одно- или многострочным
ComboBox  (поле со списком)	Этот элемент управления объединяет окно редактирования и окно списка. Используйте, когда хотите предложить пользователю выбрать значение, но при этом дать ему возможность ввести данные, отсутствующие в списке
ListBox  (список)	Отображает список значений, из которых пользователь может сделать выбор. Окна списка можно использовать, чтобы дать возможность пользователю выбрать только одно значение или же несколько
Элемент управления	Назначение
CheckBox  (флажок)	Стандартный флажок (квадратное окно, содержащее, если элемент выбран, галочку). Используйте флажки для выбора вариантов, которые не являются взаимо-исключающими
OptionButton  (переключатель)	Стандартная кнопка-переключатель (круглое окно, при выборе в центре него находится черная точка). Используйте OptionButton, когда пользователю необходимо сделать выбор между положениями «включено/выключено» или «истина/ложь». Кнопки-переключатели, как правило, объединяются вместе при помощи рамки для создания группы переключателей
ToggleButton  (выключатель)	Выключатели служат для той же цели, что и флажки, но выводят установки в виде кнопки, находящейся в «нажатом» или «отжатом» состоянии
Frame  (рамка)	Визуально и логически объединяет некоторые элементы управления (особенно флажки, переключатели и выключатели)
CommandButton  (кнопка)	Используйте кнопки для выполнения таких действий, как Cancel (Отмена), Save (Сохранить), OK и т.д. Когда пользователь щелкает по кнопке, выполняется VBA-процедура, закрепленная за данным элементом управления
TabStrip  (набор вкладок)	Этот элемент управления состоит из области, в которую вы помещаете другие элементы управления (такие как текстовые поля, флажки и т.д.) и полосы кнопок табуляции. Используйте элемент управления TabStrip для создания диалоговых вкладок, отображающих одни и те же данные в различных категориях
MultiPage  (набор страниц)	Этот элемент управления состоит из нескольких страниц. Вы можете выбрать любую из них, щелкнув по соответствующей вкладке. Используйте элемент управления MultiPage для создания диалоговых окон с вкладками
ScrollBar  (полоса прокрутки) и SpinButton  (счетчик)	Элемент управления ScrollBar позволяет выбирать линейное значение аналогично тому, как это можно сделать при помощи счетчика. Элемент управления SpinButton является специальной разновидностью текстового поля

Элемент управления	Назначение
Image  (рисунок)	Элемент управления Image позволяет вывести на форме графическое изображение. Используйте Image для вывода графических изображений в любом из следующих форматов: *.bmp, *.cur, *.gif, *.ico, *.jpg или *.wmf

Обращение к элементам управления происходит в основном через их свойства и с помощью процедур обработки событий, написанных для каждого элемента (табл. 10).

Таблица 10- Свойства стандартных элементов управления

Свойство	Где применяется	Описание
Accelerator	CheckBox, Tab, CommandButton, Label, Page, OptionButton, ToggleButton	Содержит символ, используемый в качестве быстрой клавиши вызова, элемента управления, при нажатии Alt+<клавиша быстрого вызова> происходит выбор элемента управления
BackColor	Все элементы	Число, представляющее определенный цвет фона элемента управления
Caption	CheckBox, CommandButton, Frame, Label, OptionButton, ToggleButton, Page, Tab, UserForm	Для надписи — текст, отображаемый элементом управления. Для других элементов управления — надпись, которая появляется на кнопке или вкладке или рядом с рамкой, флажком или переключателем
Cancel	CommandButton	Задаёт кнопку отмены диалогового окна. При нажатии на эту кнопку или клавишу Esc диалоговое окно исчезает. Только одна кнопка формы может иметь данное свойство
ControlTip-Text	Все элементы управления	Устанавливает текст, который отображается в виде всплывающей подсказки (ControlTip, называемой также ToolTip), когда указатель мыши помещается на элемент управления

Свойство	Где применяется	Описание
Default	CommandButton	Определяет заданную по умолчанию кнопку. Когда пользователь нажимает в процессе диалога клавишу Enter, эта кнопка ведет себя так, как если бы по ней щелкнули мышью
Enabled	Все элементы управления	Хранит значение типа Boolean, определяющее, доступен или нет элемент управления. Если Enabled имеет значение False, то элемент управления продолжает отображаться в диалоговом окне, но не может быть выбран
ForeColor	Все элементы управления	То же самое, что и BackColor, но устанавливает цвет для переднего плана элемента управления, как правило, символов текста
List	ComboBox	Массив типа variant (одно- или многомерный), представляет список, содержащийся в элементе управления
Max	ScrollBar, SpinButton	Переменная типа Long, определяющая максимальное значение счетчика, или значение, при котором полоса прокрутки находится в самом верху (для вертикальной полосы) или справа (для горизонтальной)
Min	ScrollBar, SpinButton	Переменная типа Long, определяющая минимальное значение счетчика, или значение, при котором полоса прокрутки находится в самом низу (для вертикальной полосы) или слева (для горизонтальной)
Name	Все элементы управления	Содержит имя элемента управления. Вы можете установить данное свойство только с помощью Properties Window
RowSource	ComboBox	Задаёт источник, из которого ListBox берет список объекта. В Excel VBA RowSource обычно использует диапазон рабочего листа
Selected	ListBox	Возвращает массив значений типа Boolean для списка, который допускает множественный выбор. Каждый элемент массива содержит по одному элементу, соответствующему каждому пункту списка. Если значение элемента в массиве selected равно True, то соответствующий пункт списка выбран

Свойство	Где применяется	Описание
TabIndex	Все элементы управления	Число, указывающее положение элемента управления в порядке табуляции (может иметь значение от 0 до значения, равного количеству элементов управления на форме)
TabStop	Все элементы управления	Значение типа Boolean, указывающее, может ли элемент управления быть выбран клавишей Tab. Если значение TabStop равно False, вы тем не менее можете щелкнуть на элементе и таким образом его выбрать
Value	Все элементы управления	Значение текущих установок элемента управления: текст в текстовом поле, какие выбраны флажки и переключатели, индекс выбранного раздела списка или число, указывающее текущее положение полосы прокрутки или счетчика
Visible	Все элементы управления	Значение типа Boolean, указывающее, является ли элемент управления видимым

## 2.2. Использование форм

### Создание VBA-программ

Используя формы, можно достаточно полно изучить возможности работы со всеми типами данных и их взаимодействия с учетом того, что значения переменных будут определяться не программно (примеры 1, 2, 3), а вводом через текстовые поля формы, т. е. при непосредственной работе пользователя с программой.

Рассмотрим создание программ, которые взаимодействуют непосредственно с создаваемыми формами и где будет использоваться весь материал, изученный в главе 1.

*Пример 4.* Создать форму, в которой при вводе имени в текстовое поле после нажатия кнопки *ОК* выдается приветственное сообщение в метку в виде: «Имя, привет! Сегодня – дата и время запуска программы».

Программа, считывая значение с текстового поля, выводит в соответствующий элемент управления данное значение и дополнительно использует функцию вывода времени и даты запуска программы.

#### Технология выполнения

Любая разработка программы на VBA будет сопровождаться разработкой формы, которая непосредственно связана с создаваемой программой. Поэтому на данном примере будет подробно рассмотрен порядок выполнения работы.

##### *1-й шаг. Проектирование программы-примера*

Программа-пример должна будет открывать на экране новое окно с показанным в нем приветствием, а также датой и временем (сообщением). Окно будет оставаться на экране до тех пор, пока пользователь не щелкнет на кнопке *ОК*.

Имея подробное описание задачи, можно определить те элементы, из которых должна состоять форма, взаимодействующая с разрабатываемой программой. По условиям примера программа имеет одно пользовательское диалоговое окно, поэтому необходимо создать одну форму (UserForm). Для формы потребуются два элемента управления – надпись для сообщения и кнопка для команды *ОК*. Нужно будет также создать программный код для двух процедур: одной – для надписи, в которую нужно поместить сообщение, а другой – для выхода из программы, когда пользователь щелкнет на кнопке *ОК*.

##### *2-й шаг. Реализация проекта*



Для выполнения данного шага выполните следующие действия.

1. Активизируйте приложение Word, сохраните документ под соответствующим именем (Время) и перейдите в редактор VBA.
2. Щелкните правой кнопкой мыши в окне проекта по пункту Project (Время), выберите пункт *Insert + UserForm* (появится новая форма UserForm с панелью элементов управления).
3. Расположите на форме следующие элементы: TextBox, Button1, Button2, Label1, Label2 (см. рис. 13).

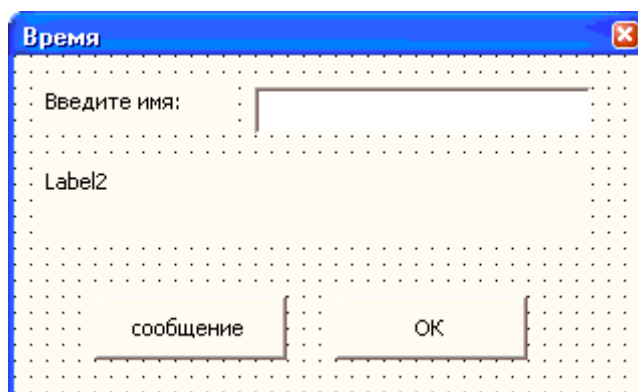


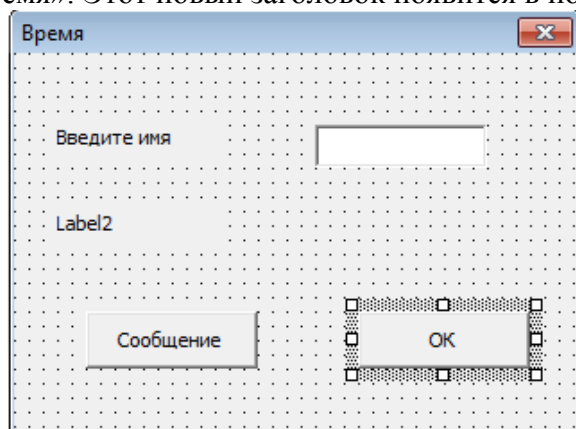
Рис. 13. Форма для примера 4 в режиме конструктора

#### *Добавление надписи в форму*

Чтобы поместить надпись в пользовательскую форму, выполните следующее.

- Убедитесь, что форма активна, щелкнув по ней. Панель элементов управления видна только тогда, когда форма активна.
- Щелкните на кнопку A (метка) панели элементов управления.
- Поместите указатель мыши в форму, где необходимо расположить сообщение.
- В окне свойств (*Properties* – левая нижняя панель окна) выделите свойство *Caption* и наберите строчку «Введите имя», во второй метке данное свойство должно быть пустым, чтобы при запуске программы надпись Label2 была невидима.

4. Измените заголовок самой формы. Для этого щелкните на полосе заголовка окна формы. В изменившемся при этом окне свойств найдите свойство *Caption* и измените его на «Время». Этот новый заголовок появится в полосе заголовка формы.



#### *Добавление программного кода*

Для нашей программы требуется создать две процедуры, и они связаны с событиями, возникающими в процессе выполнения программы. Первая процедура должна при нажатии на кнопку *Сообщение* отобразить нужное сообщение, а вторая – завершить выполнение программы, когда кто-нибудь щелкнет на кнопке *ОК*.

5. Щелкните дважды по кнопке ОК. В появившемся при этом окне программного кода появится заготовка процедуры. Первой строкой созданного программного кода будет Private Sub CommandButton2\_Click().

В любой VBA-процедуре первая строка программного кода определяет тип процедуры (в данном случае это процедура типа Sub, т. е. подпрограмма) и имя процедуры. Private и Sub относятся к ключевым словам VBA, т. е. к словам и символам, которые являются частью языка VBA. В данном случае VBA предлагает для процедуры имя CommandButton2\_Click, которое представляет собой комбинацию имени кнопки и типа события.

Последней строкой автоматически генерируемого программного кода будет End Sub.

Такой строкой должны заканчиваться все процедуры типа Sub. Эта строка сообщает VBA о том, что выполнение процедуры завершено.

Для выполнения первой процедуры (закрытия формы) необходимо прописать программный код в этой заготовке:

*Unload Me*

Оператор Unload убирает указанный объект из памяти. Здесь это объект с именем Me, имеющим в VBA специальный смысл. В данном случае оно означает форму и весь ее программный код.

6. Создайте обработчик события для кнопки «Сообщение», для чего дважды щелкните по созданной кнопке и пропишите код:

*Dim ima As String*

*ima = TextBox1.Text*

*Label2.Caption = ima & ",привет! Сегодня " & Format(Now, «dddddd, hh ч. mm мин.»)*

Вторая процедура, которая должна отображать на экране сообщение, чуть сложнее первой. Первая из напечатанных строк

*Dim ima As String*

создает переменную с именем ima и определяет ее как строковую, что означает последовательность текстовых символов.

Вторая строка данной переменной присваивает строковое значение, введенное в элемент TextBox1 (текстовое окно может «читать» только текстовое значение). Третья строка выводит сообщение в расположенную на форме метку Label2, для чего устанавливает свойству Caption программно-строковое значение. Функция Format выдает дату и время.

*3-й шаг. Тестирование программы*

Чтобы запустить программу из редактора Visual Basic, выполните следующее.

1. Щелкните либо в окне формы, либо в окне программного кода, чтобы соответствующее окно стало активным.

2. Прокомпилируйте программу: меню + debug + compile project.

3. Запустите программу на выполнение (F5).

После небольшой задержки окно вашей программы появится на фоне вашего VBA-приложения (а не редактора Visual Basic). Если все в порядке, на фоне приложения (Word или Excel) появится созданная форма в рабочем состоянии (рис. 14).

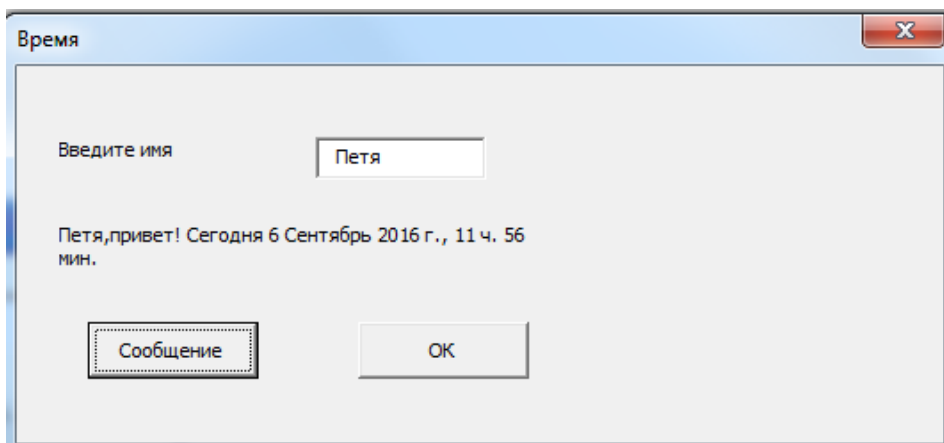


Рис. 14. Форма примера 4 в рабочем состоянии

- Это важно!

Если при запуске разработанной формы появляются сообщения вида (рис. 15 или рис. 16) о слишком большой защите приложения от макросов, т. е. вмешательства извне, необходимо выполнить следующие действия:

- 1) закрыть редактор VBA;

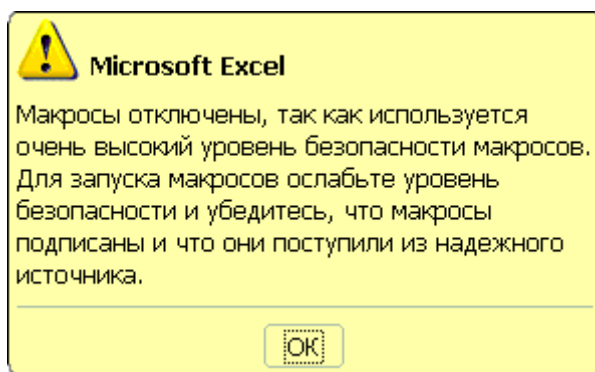


Рис. 15. Предупреждение 1-го вида

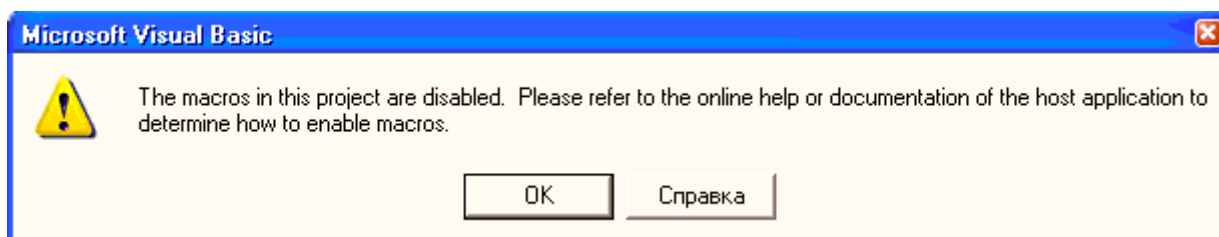


Рис. 16. Предупреждение 2-го вида о высокой защите приложений

2) в приложении Excel или Word (там, где происходит работа в текущий момент) выполнить команду: параметры Office + Центр управления безопасностью + параметры центра управления безопасностью + параметры макросов;

4) установить включить все макросы, так как создаваемая форма не несет никакой угрозы для операционной системы вашего компьютера (рис. 17);

5) закрыть приложение (Word или Excel) и запустить его вновь.

Рассмотрим простейшие примеры работы с циклами.

*Пример 5.* Используя инструкцию case, создать программу, которая в зависимости от введенного значения переменной *a* производит различные вычисления с переменными *b* и *c*. Если значение переменной *a* не совпадает с программными, то выдается сообщение «Введено не то значение».

Все переменные вводятся в текстовые поля формы. При нажатии на кнопку «Результат» происходят выбор действия и вывод полученного значения в специальную метку формы.

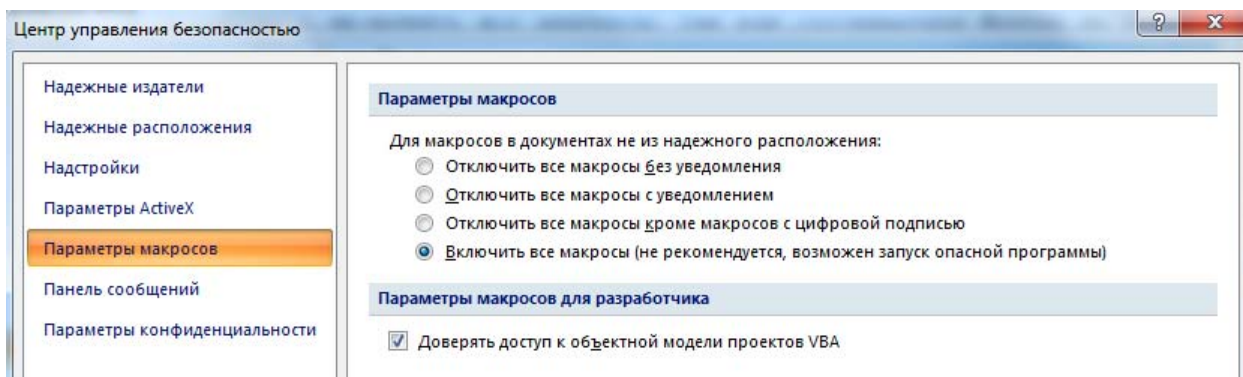


Рис. 17. Включение все макросов

### Технология выполнения

1. Активизируйте приложение Word, сохраните документ под именем Case.
2. Перейдите в редактор VBA и создайте форму (рис. 18).
3. Пропишите обработчик кнопки «Результат».

```
Dim a, b, c, d As Integer
Private Sub CommandButton1_Click()
    a = Val (TextBox1.Text)
    b = Val (TextBox2.Text)
    c = Val (TextBox3.Text)
    Select Case a
        Case 5
            d = b + c
            Label4.Caption = "Результат: d=" & d
        Case 0
            d = -b - c
            Label4.Caption = "Результат: d=" & d
        Case 10
            d = b * c
            Label4.Caption = "Результат: d=" & d
        Case Else
            Label4.Caption = "Введено не то значение"
    End Select
End Sub
```

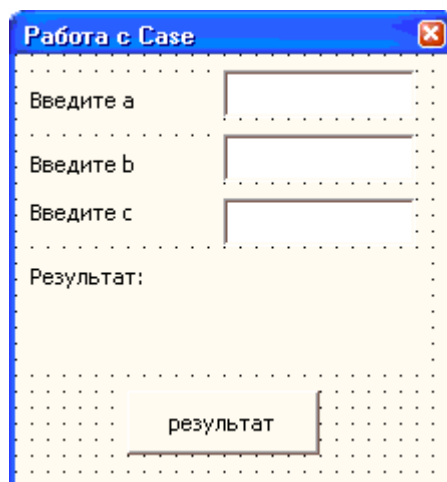


Рис. 18. Форма примера 5 в режиме конструктора



Для того чтобы программа работала корректно, необходимо перевести все текстовые значения, которые вводятся в текстовые поля, в числовые. Для этого прописывают в коде преобразование: `a = Val(TextBox1.Text)` и т. д. После чего программа уже работает с числовыми значениями.

Если программа в итоге выдает значение одного типа (числовое или строковое), то строку вывода результата можно прописывать по окончании всего блока Case. В данном примере сначала выводятся числовые значения (d), а затем строковое («Введено не то значение»). Поэтому вывод результата необходимо предусмотреть в каждой ветке Case.

Результаты работы показаны на рис. 19.

*Пример 6.* Создать программу, которая, используя пользовательское диалоговое окно (форму), выполняет следующие действия: при вводе трех переменных в текстовые поля она считывает данные и сравнивает с первой переменной a. Результат выдается в метку на форме (рис. 20).

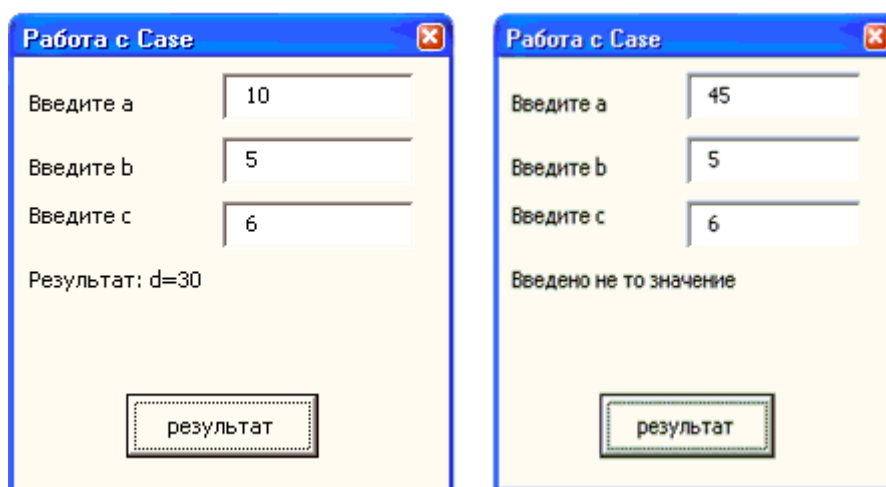


Рис. 19. Некоторые результаты работы формы примера 5

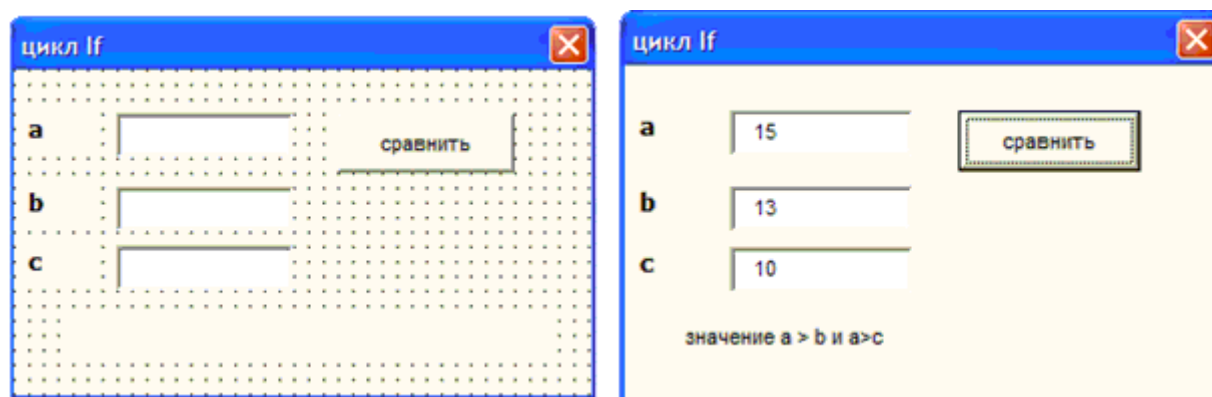


Рис. 20. Форма примера 6 в режиме конструктора и в рабочем состоянии

Листинг примера 6

```

Private Sub CommandButton1_Click()
Dim a, b, c As Integer
a = Val(TextBox1.Text)
b = Val(TextBox2.Text)
c = Val(TextBox3.Text)
If a > b And a > c Then
Label4.Caption = "Значение a > b и a > c"
Else
Label4.Caption = "Значение a не всегда больше b и c"
End If
End Sub

```

*Пример 7.* Создать программу, которая, используя форму, выполняет следующие действия: при вводе переменной в текстовое поле она считывает данное значение a, после чего организует цикл for с шагом, равным 5, где при каждом шаге значение переменной b становится равным значению переменной a плюс шаг изменения. Итоговое значение c суммирует полученное значение b и введенное значение a. Результат выдается в метку на форме (рис. 21).

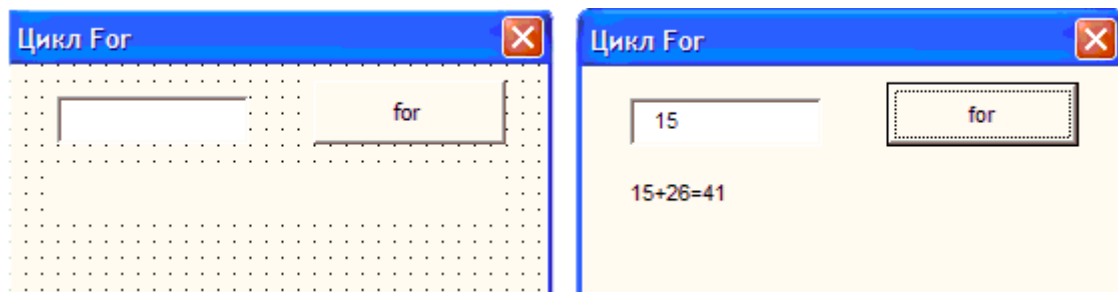


Рис. 21. Форма примера 7 в режиме конструктора и в рабочем состоянии

Листинг примера 7

```

Dim a As Variant
Dim b As Integer
Dim c As Integer
Private Sub CommandButton1_Click()
a = Val(TextBox1.Text)
For i = 1 To 12 Step 5
b = a + i
c = a + b
Next i
Label1.Caption = a & "+" & b & "=" & c
End Sub

```

*Пример 8.* Создать программу, которая выполняет следующие действия: организованный цикл for each присваивает переменной b, объявленной в программе, последнее значение массива, также определенного программой. После этого происходят вычисления, предложенные в программе:

- в метку label2 выдается результат увеличения полученной переменной на значение первого элемента массива;
- в метку label3 выдается результат увеличения полученной суммы на значение второго элемента массива;
- в метку label4 выдается результат увеличения полученной суммы на значение третьего элемента массива;
- в метку label6 выдается результат увеличения полученной суммы на значение последнего элемента массива.

Результат выдается для наглядности в различные метки на форме (рис. 22).

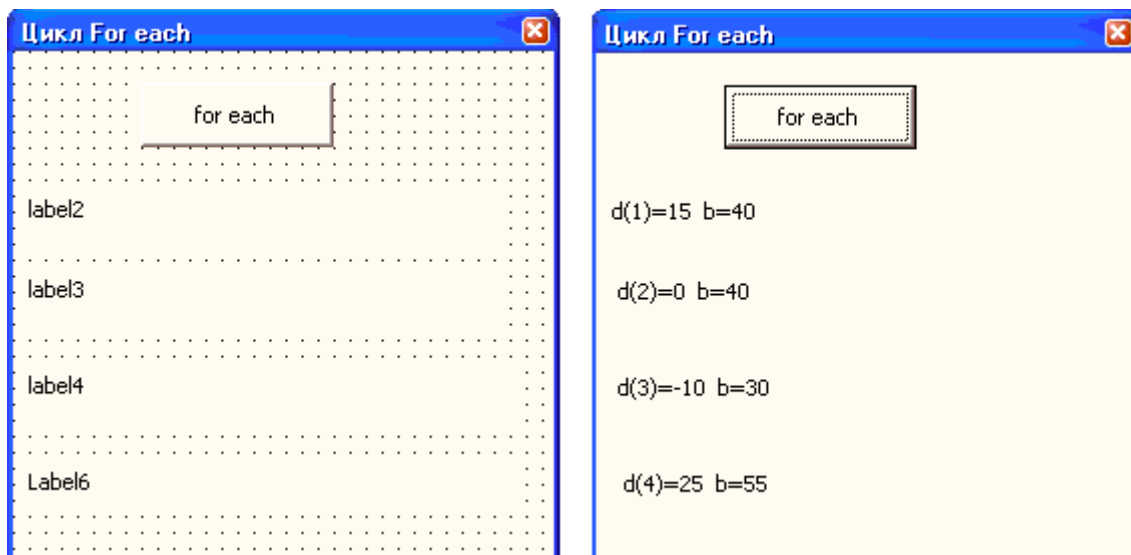


Рис. 22. Форма примера 8 в режиме конструктора и в рабочем состоянии

#### Листинг примера 8

```

Dim b As Variant
Private Sub CommandButton1_Click()
Dim d(1 To 4) As Variant
d(1) = 15
d(2) = 0
d(3) = -10
d(4) = 25
For Each b In d
b = d(1) + b
Label2.Caption = "d(1)=" & d(1) & " b=" & b
b = d(2) + b
Label3.Caption = " d(2)=" & d(2) & " b=" & b
b = d(3) + b
Label4.Caption = " d(3)=" & d(3) & " b=" & b
b = d(4) + b
Label6.Caption = " d(4)=" & d(4) & " b=" & b
Next b
End Sub

```

*Примечание.* Как говорилось выше, For Each ... Next не использует счетчик цикла. Циклы For Each ... Next выполняются столько раз, сколько имеется элементов в определенной группе, такой как коллекция объектов или массив. Другими словами, цикл For Each ... Next выполняется один раз для каждого элемента в группе. Вследствие чего данный цикл используется в основном в специфических действиях, таких как поиск необходимого листа в коллекции объектов и т. д., и не используется при решении простых задач.

Задачи на закрепление материала

*Пример 9.* Создать программу, которая, используя данные, считанные из текстовых полей формы, после нажатия на кнопку выводит сообщение:

«Здравствуй, *введенное имя*, студент группы *номер группы* специальности *название специальности!*»,

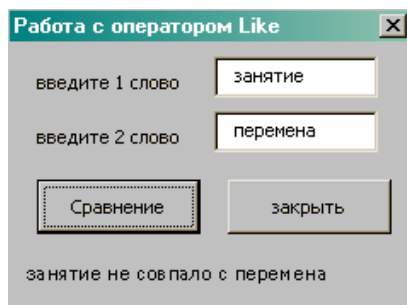
где *введенное имя* – значение из первого текстового поля;

*номер группы* – значение из второго текстового поля;

*название специальности* – значение из третьего текстового поля.

*Пример 10.* Создать программу, которая, используя данные, считанные из текстовых полей формы, выполняет следующие действия: если введенный текст одинаков, то выдается сообщение «значение1 совпало с значением2», если введенный текст неодинаков, то выдается сообщение «значение1 не совпало с значением2», где *значение1* и *значение2* – введенные слова.

*Примечание.* При разработке программы лучше использовать оператор *Like* (рис. 23).



Работа с оператором Like

введите 1 слово: занятие

введите 2 слово: перемена

Сравнение      закрыть

занятие не совпало с перемена

Рис. 23. Форма примера 10 в рабочем режиме

## Лабораторная работа №3. Массивы, процедуры, функции

### 3.1. Организация массивов

#### Одномерные массивы

*Массив (array)* – это коллекция переменных, которые имеют общие имя и базовый тип. Массив является удобным способом хранения нескольких связанных элементов данных. Все элементы данных, сохраняемых в массиве, должны иметь один и тот же тип.

Наименее сложный массив – это просто список элементов данных; такого рода массив называется *простым*, или *одномерным*, массивом. Подобный массив можно представить в виде очереди, где каждому элементу очереди присваивается не только порядковый номер (место в очереди), но и его конкретное значение (имярек).

Чтобы создать массив, нужно определить: его имя, количество элементов (размер массива), тип данных, которые будут храниться в массиве.

Массивы создаются при помощи оператора Dim:

Dim ИмяМассива (размер) As ТипДанных

Если вместо слова Dim набрать слово Public, будет создан массив, областью видимости которого станет вся программа.

Пример:

*Dim МойМассив(3) As Integer*

Создан массив по имени МойМассив, состоящий из четырех элементов и способный хранить значения типа Integer.

Так как отсчет элементов начинается с нулевого номера, то ставится цифра 3.

Другая версия задания массива: указать первый и последний номера элементов массива:

*Dim МойМассив (1 To 3) As Integer*

Элементы созданного массива не содержат никаких данных. Чтобы сохранить в массиве какое-нибудь значение, нужно указать, какому элементу оно должно быть присвоено. Предположим, создан массив, который может содержать в себе до пяти строк:

Dim Сотрудник(4) As String

Если необходимо первому элементу массива присвоить значение Иван Никитин, надо прописать такой код:

Сотрудник(0) = «Иван Никитин»

В большинстве программ при создании массива сразу же инициализируют его, присвоив каждому элементу нулевое значение или пустую строку. Это можно сделать, используя цикл for next, например:

*Dim сотрудник(4) As String*

*Dim I As Integer*

*For I = 0 To 4*

*Сотрудник (i) = ""*

*Next I*

Значения элементов массива можно присваивать другим переменным, например:

*Dim сотрудник(4) As String*

*Dim отпуск As String*

*Сотрудник(3) = «Иван Никитин»*

*отпуск = сотрудник(3)*

Здесь:

1) первой строкой создается массив Сотрудник, состоящий из пяти текстовых значений;

2) второй строкой создается текстовая переменная, именуемая отпуск;

3) третьей строкой четвертому элементу массива (которому соответствует третий порядковый номер) присваивается значение Иван Никитин;

4) в четвертой строке переменной отпуск присваивается значение элемента массива Сотрудник, которому соответствует третий порядковый номер.

Обычно элементы массива содержат значения, относящиеся к одному типу данных, например только строки или только целые числа. Если же необходимо, чтобы в массиве содержались данные разных типов, при создании массива укажите тип данных Object:

```
Dim МойМассив as object
```

Элементы такого массива могут содержать значения разных типов:

```
мойМассив(0) = «Спрут»
```

```
мойМассив(1) = 56
```

```
мойМассив(2) = 3.1415
```

### Двумерные массивы

Порядок создания двумерного массива тот же, что и одномерного, с той лишь разницей, что, указывая его размер, нужно указать два значения – строки и столбцы:

```
Dim ДвухММассив (Строки, Столбцы) As ТипДанных
```

Размер трехмерного массива будет определяться тремя числами и т. д.:

```
Dim ТрехММассив (X, Y, Z) As ТипДанных
```

При создании массивов, в том числе и многомерных, для хранения значения каждого элемента выделяется оперативная память (даже если это нулевые значения или пустые строки). Таким образом, создавая большой массив, происходит резкое уменьшение объема свободной памяти, что может негативно отразиться на работе программы. Поэтому создавать многомерные массивы следует лишь по мере необходимости. Подобные массивы называются *статическими (static)*, потому что число элементов в массиве не меняется.

Выбор размера массива может быть затруднен, если неизвестно, сколько данных будет введено в массив, или если объем данных, собираемых для массива, значительно меняется. Для подобных ситуаций VBA поддерживает особый тип массивов, называемый *динамическим (dynamic)* массивом.

Динамические массивы создаются с помощью оператора Dim, Private, Public или Static, причем список размерностей опускается, затем их размер устанавливается с помощью оператора ReDim во время выполнения процедуры.

Оператор ReDim имеет следующий синтаксис:

```
ReDim [Preserve] varname (subscripts) [As type] [, varname (subscripts) [As type]],
```

где необязательное ключевое слово *Preserve* приводит к тому, что VBA сохраняет данные в имеющемся массиве, когда изменяется размер массива с помощью ReDim;

*varname* – имя существующего массива;

*subscripts* – измерения массива (синтаксис для оператора *subscripts* в операторе ReDim такой же, как для оператора Dim);

*type* – любой тип VBA или определенный пользователем тип.

Необходимо использовать отдельный оператор *As type* для каждого массива, который вы определяете.

Примеры:

1) Dim Month() As String – объявляет динамический массив Month;

2) ReDim Month(1 to 30) – изменяет размер массива до 30 элементов;

3) ReDim Month(31) – изменяет размер массива до 31 элемента;

4) ReDim Preserve Month(1 to 31) – изменяет размер массива до 31 элемента, сохраняя содержимое;

5) Dim Table() As Integer – объявляет динамический массив;

6) ReDim Table(3, 15) – делает массив двумерным;

7) ReDim Table(4, 20) – изменяет размер двумерного массива;

8) ReDim Preserve Table(4, 25) – только изменяет последний размер массива;

9) Dim Mas as Variant – объявляет переменную типа Variant;

10) ReDim Mas(20) As Integer – создает массив 20 целых чисел в Variant.

**Выводы:**

1) можно изменять только последнее измерение многомерного массива, когда используется ключевое слово *Preserv*;

2) можно использовать *ReDim* для создания типизированного массива внутри переменной типа *Variant*.

Массив в программе можно также определить поэлементно. Например, следующий код

```
Dim B(l to 2, 1 to 2) as single
```

```
B(1,1)=2
```

```
B(1,2)=5
```

```
B(2,1)=4
```

```
B(2,2)=3
```

создает двумерную таблицу

$$\begin{pmatrix} 2 & 5 \\ 4 & 3 \end{pmatrix}$$

При работе с массивами бывает полезно применять следующие функции и процедуры.

1. *Array* (списокАргументов)

Создает массив типа *Variant*. Аргумент в скобках представляет разделенный запятыми список значений, присваиваемых элементам массива.

Пример:

```
Dim День As Variant
```

```
День=Array(«Пн», "вт", "ср", ....)
```

2. VBA имеет две функции, которые отслеживают верхний и нижний индексы предела массива, – функции *Lbound* и *ubound*. Эти функции возвращают нижнее и верхнее граничные значения индексов статического или динамического массива.

Синтаксис:

```
Lbound (имяМассива [, размерность])
```

```
Ubound (имяМассива [, размерность]),
```

где *ИмяМассива* – имя переменной массива;

*Размерность* – целое число, указывающее размерность массива, нижнюю или верхнюю границу которой возвращает функция. Для первой размерности следует указать 1, для второй – 2 и т. д. Если аргумент размерность опущен, подразумевается значение 1.

3. Использование оператора *Erase* для очистки или удаления массивов.

Оператор *Erase* позволяет выполнять одну из двух задач в зависимости от того, каким массивом манипулирует пользователь – статическим или динамическим. В случае статических массивов *Erase* позволяет очищать все элементы массива, в основном переустанавливая массив в то же самое состояние, какое он имел, когда VBA создавал его в оперативной памяти. В случае динамических массивов *Erase* позволяет полностью удалять массив и его содержимое из оперативной памяти.

VBA удаляет из памяти массивы, объявляемые локально в процедуре (так же, как и любые другие локальные переменные), каждый раз, когда процедура прекращает выполняться. Однако массивы, объявляемые на модульном уровне, существуют, пока любая процедура в этом модуле выполняется. Если программа большая, можно восстановить ресурс памяти, используемой динамическими массивами модульного уровня. Оператор *Erase* позволяет делать именно это.

Оператор *Erase* имеет следующий синтаксис:

```
Erase array1 [, array2, ...]
```

Здесь *array1* и *array2* представляют любое допустимое имя массива VBA.

Оператор *Erase* удаляет из памяти динамические массивы, освобождая область памяти, ранее используемую этим массивом. При удалении динамического массива с помощью оператора *Erase* необходимо повторно создать массив с помощью оператора

ReDim перед тем, как можно будет использовать этот определенный динамический массив снова. При попытке доступа к элементам в динамическом массиве, для которого был использован оператор Erase, без его переопределения, VBA отображает сообщение об ошибке.

Обычно в VBA используются массивы с нулевой базой. В системе нумерации с нулевой базой индекс для первого элемента в любом измерении массива является равным 0; массив с 10 элементами имеет индексы от 0 до 9.

Было бы гораздо удобнее, если бы элементы массива нумеровались начиная с 1, а не с 0. VBA позволяет задавать начальное число для элементов массива, используя директиву компилятора Option Base для указания того, должна ли нумерация индексов начинаться с 0 или с 1.

Директива компилятора Option Base имеет следующий синтаксис:

Option Base 0 | 1

Если оператор Option Base не используется, VBA начинает нумерацию индексов массива с 0 (по умолчанию). Необходимо помещать оператор Option Base в область объявлений модуля перед объявлениями любых переменных, констант или процедур. Нельзя помещать оператор Option Base внутри процедуры. Можно иметь только один оператор Option Base в модуле; оператор Option Base влияет на все массивы, объявляемые в модуле, независимо от того, являются ли они локальными в процедуре или объявляются на модульном уровне.

Например:

Option Base 0 'установка по умолчанию с нуля

Option Base 1 'индексы массивов начинаются с 1

*Пример 11.* Создать программу, организующую три двумерных массива. Первые два массива определены поэлементно в программе. Третий массив А организуется путем суммирования соответствующих членов массивов В и С (рис. 24).

Результаты организации массивов выведены в соответствующие метки на форме после нажатия на кнопку *Массив*.

Листинг примера 11

```
Private Sub CommandButton1_Click()  
    Dim B(1 To 2, 1 To 2) As Integer  
    Dim c(1 To 2, 1 To 2) As Integer  
    Dim A(1 To 2, 1 To 2) As Integer  
    B(1, 1) = 5  
    B(1, 2) = 4  
    B(2, 1) = 8  
    B(2, 2) = 10  
    c(1, 1) = 0  
    c(1, 2) = 1  
    c(2, 1) = 5  
    c(2, 2) = 10  
    For i = 1 To 2  
        For j = 1 To 2  
            A(i, j) = B(i, j) + c(i, j)  
        Next j  
    Next i  
    Label1.Caption = "a(1,1)=" & A(1, 1) & " a(1,2)=" & A(1, 2) & " a(2,1)=" & A(2, 1) & " a(2,2)=" & A(2, 2)  
    Label2.Caption = "b(1,1)=" & B(1, 1) & " b(1,2)=" & B(1, 2) & " b(2,1)=" & B(2, 1) & " b(2,2)=" & B(2, 2)  
    Label3.Caption = "c(1,1)=" & c(1, 1) & " c(1,2)=" & c(1, 2) & " c(2,1)=" & c(2, 1) & " c(2,2)=" & c(2, 2)  
End Sub
```



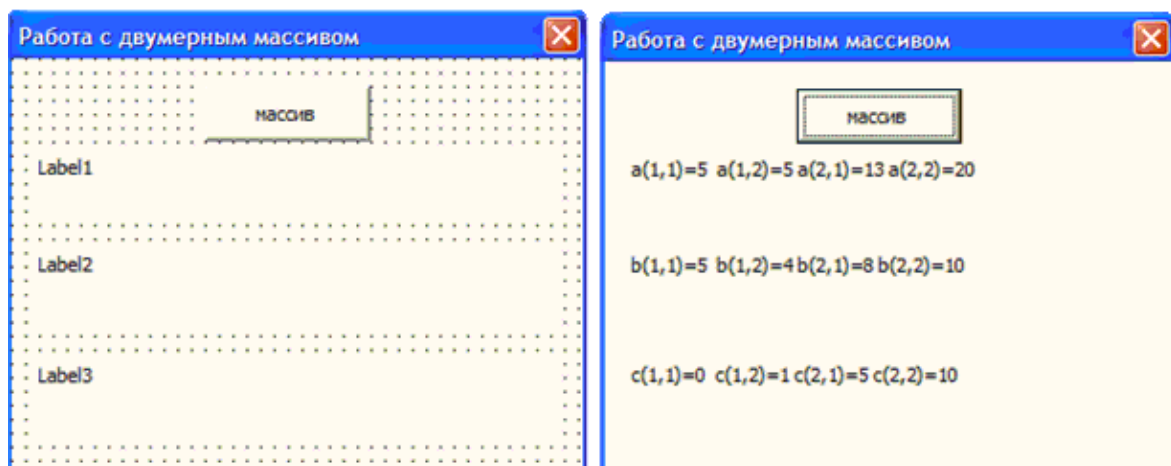


Рис. 24. Форма примера 11 в режиме конструктора и в рабочем состоянии

*Пример 12.* Создать программу, создающую два двумерных массива: один массив вводом числовых элементов в соответствующие текстовые поля формы, второй – вводом четырех произвольных фамилий в соответствующие текстовые поля формы (рис. 25, 26). В результате все элементы первого массива увеличиваются на 10 и выводятся в соответствующую метку на форме. Элементы же второго строкового массива организуют вывод предложений типа: работник фамилия *Иванов*, где *Иванов* (например) берется из строкового массива, введенного в соответствующие текстовые поля. Данные результаты получаются после нажатия на кнопку *Вывод* пользовательского диалогового окна.

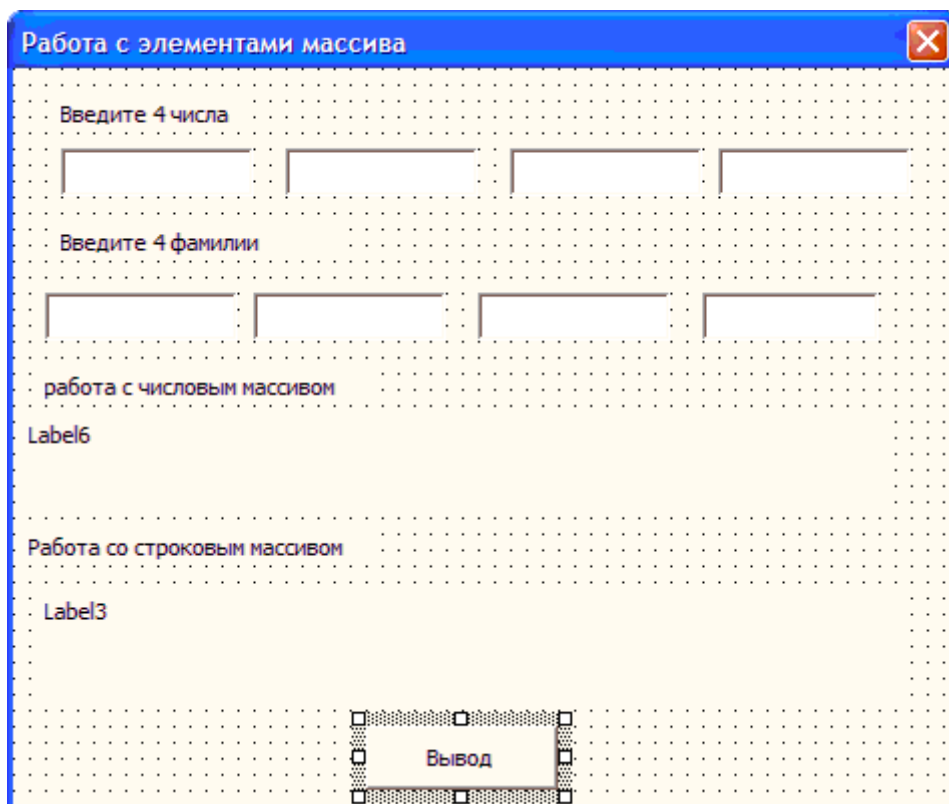


Рис. 25. Форма примера 12 в режиме конструктора

Рис. 26. Форма примера 12 в рабочем режиме

### Листинг примера 12

```
Private Sub CommandButton1_Click()
    Dim B(1 To 2, 1 To 2) As String
    Dim c(1 To 2, 1 To 2) As String
    Dim A(1 To 2, 1 To 2) As String
    Dim d(1 To 2, 1 To 2) As Integer
    Dim k(1 To 2, 1 To 2) As Integer
    B(1, 1) = TextBox5.Text
    B(1, 2) = TextBox6.Text
    B(2, 1) = TextBox7.Text
    B(2, 2) = TextBox8.Text
    d(1, 1) = Val(TextBox1.Text)
    d(1, 2) = Val(TextBox2.Text)
    d(2, 1) = Val(TextBox3.Text)
    d(2, 2) = Val(TextBox4.Text)
    For i = 1 To 2
        For j = 1 To 2
            k(i, j) = d(i, j) + 10
        Next j
    Next i
    For i = 1 To 2
        For j = 1 To 2
            c(i, j) = "фамилия" + B(i, j)
        Next j
    Next i
    For i = 1 To 2
        For j = 1 To 2
            A(i, j) = "работник" + c(i, j)
        Next j
    Next i
    Label3.Caption = "a(1,1)=" & A(1, 1) & " a(1,2)=" & A(1, 2) & " a(2,1)=" & A(2, 1) & " a(2,2)=" & A(2, 2)
    Label6.Caption = "k(1,1)=" & k(1, 1) & " k(1,2)=" & k(1, 2) & " k(2,1)=" & k(2, 1) & " k(2,2)=" & k(2, 2)
End Sub
```

## 3.2. Работа с различными типами данных

Тип, определяемый пользователем

VBA позволяет пользователю определять свои собственные типы данных. Определенный пользователем тип нужен, когда одной переменной необходимо обозначить несколько связанных по смыслу элементов данных, причем эти элементы данных могут быть разных типов. Пример структурного типа приведен на рис. 27. Тип Book состоит из трех элементов: *Title* (название книги) имеет тип String, *Content* – динамический массив строкового типа, содержащий название глав книги, *Author* (автор книги), который, в свою

очередь, тоже является структурным типом, состоящим из двух простых элементов – *Name* (имя) и *Birthday* (день рождения).

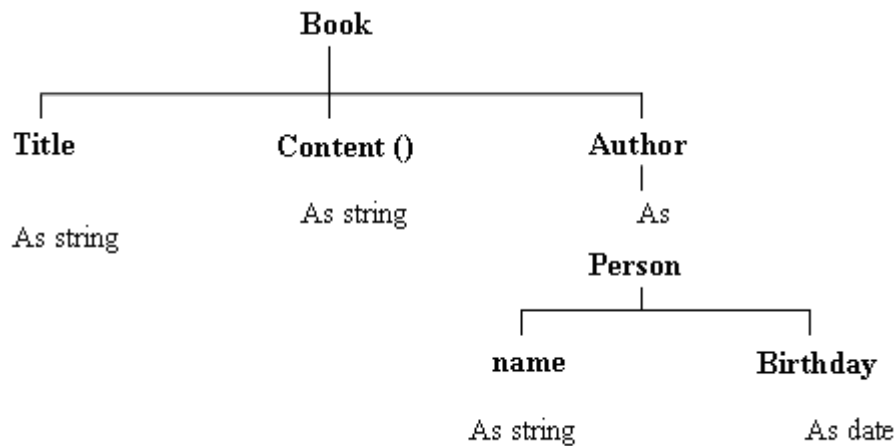


Рис. 27. Пример структурного типа, определяемый пользователем

Для объявления такого типа данных используется оператор Type:

Type <имяТипа>

...<имяЭлемента1> [[(<размер1>)] As <типДанных1>

...<имяЭлемента2> [[(<размер2>)] As <типДанных2>

End Type

Элементами типа могут быть простые переменные и массивы встроенных типов, а также переменные и массивы других определенных пользователем типов. Типы Book и Person (см. рис. 27) могут быть объявлены следующим образом:

Type Person

Name As String

Birthday As Date

End Type

Type Book

author As Person

Title As String

Content () as String

End Type

Объявление переменных структурного типа выполняется так же, как и обычных переменных:

Dim MyBook As Book, Editor As Person

Обращение к элементу структурного типа выполняется следующим образом:

MyBook.Title = «Учебник»

ReDim MyBook.Content (0 to 10)

MyBook.Content(0) = «От автора»

Перечисляемый тип

Еще одним видом структурного типа данных является *перечисляемый* тип. Элементами перечисляемого типа являются все его значения. Определяется перечисляемый тип с помощью оператора Enum:

Enum <имяТипа> <имяЗначения1> [=<Константа>] <имяЗначения2>

[=<Константа>]

End Enum

По умолчанию все значения типа перенумеровываются целыми числами, начиная с 0, но можно самостоятельно указать константное целое значение для значений типа.

Примером перечисляемого типа данных является встроенный логический тип Boolean.

Enum Boolean False

True End Enum

Над каждым типом данных определено некоторое множество простейших действий, называемых *операциями*. Язык VBA имеет большое количество *встроенных функций* для работы с каждым типом данных.

Приведение и преобразование типов

*Приведением* называется автоматическое преобразование значения одного типа данных в эквивалентное значение другого типа в процессе выполнения операций с данными. Приведение выполняется, если *операнды* (данные, участвующие в операции) имеют разные типы. При этом результат операции будет иметь тот тип, к которому приводится один из операндов. Например, складываются два числа – целое (Integer) и вещественное (Double). В процессе этой операции целое число приводится к вещественному (Double), и результат будет иметь тип Double. Обратное преобразование (Double в Integer) может привести к потере данных. На рис. 28 приведена схема, показывающая, значения каких типов к каким типам приводятся без потери информации.

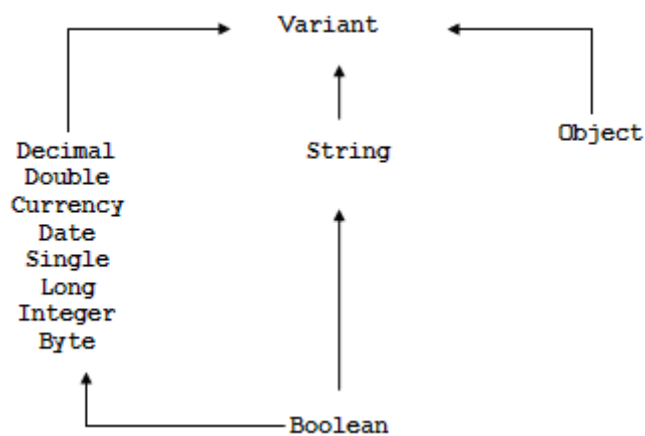


Рис. 28. Схема преобразования данных без потерь

VBA имеет также набор функций, которые можно использовать для явного преобразования типов данных в операциях. Эти функции приведены в табл. 11.

Таблица 11 Функции преобразования типов

Функция	Возвращает действие	Возвращаемый тип
Asc(S)	Возвращает число кода символа, соответствующее первой букве строки S. Буква «А», например, имеет код символа 65	Integer
CBool(N)	Возвращает Boolean-эквивалент численного выражения N	Boolean
Format(E, S)	Возвращает строку, содержащую значение, представленное выражением E, в формате в соответствии с инструкциями, содержащимися в S	String
Str(N)	Возвращает строку, эквивалентную численному выражению N	String
Val(S)	Возвращает численное значение, соответствующее числу, представленному строкой S, которая должна содержать только цифры и одну десятичную точку, иначе VBA не может преобразовать ее в число. Если VBA не может преобразовать строку в S, то функция Val возвращает 0	Variant
CByte(E)	Возвращает численное значение типа Byte (от 0 до 255); E — любое допустимое численное или строковое выражение, которое может быть преобразовано в число	Byte
CCur(E)	Возвращает численное значение типа Currency; E — любое допустимое численное или строковое выражение, которое может быть преобразовано в число	Currency
CDate(E)	Возвращает значение типа Date. E может быть любым допустимым выражением (строкой или числом), представляющим дату в диапазоне 1/1/100–12/31/9999 включительно	Date
CDbl(E)	Возвращает численное значение типа Double; E — любое допустимое численное или строковое выражение, которое может быть преобразовано в число	Double

Функция	Возвращает действие	Возвращаемый тип
CInt(E)	Возвращает численное значение типа Integer; E — любое допустимое численное или строковое выражение, которое может быть преобразовано в число	Integer
CLng(E)	Возвращает численное значение типа Long; E — любое допустимое численное или строковое выражение, которое может быть преобразовано в число	Long
CSng(E)	Возвращает численное значение типа Single; E — любое допустимое численное или строковое выражение, которое может быть преобразовано в число	Single
CStr(E)	Возвращает значение типа String; E — любое допустимое численное или строковое выражение	String
CVar(E)	Возвращает значение типа Variant; E — любое допустимое численное или строковое выражение	Variant
Chr(N)	Возвращает строку из одного символа, соответствующего коду символа N, который должен быть числом между 0 и 255 включительно. Код символа 65, например, возвращает букву «А»	Символ

#### Операции со строками

Для данных типа String существует только одна операция – конкатенация (объединение). Например, результатом операции конкатенации трех строковых значений

«Петр» & " " & «Иванович» будет строка «Петр Иванович». Возможно также использование другого оператора для операции конкатенации, например: «десяти» + «тысячник». Разница между этими выражениями состоит в том, что в первом случае операндами могут быть значения любого типа (они просто будут преобразовываться в строковые), а во втором – оба операнда должны иметь тип String.

Для работы со строками существует большое количество функций (табл. 12).

Таблица 12 Функции работы со строками

Функция	Описание	Пример
Len(str)	Определяет длину строки	Из a = len("Персонажи") следует a=9
Left (<строка>, <длина>)	Выделяет из аргумента <строка> указанное количество символов слева	Left("1234string",4) = "1234"
Right (<строка>, <длина>)	Выделяет из аргумента <строка> указанное количество символов справа	Right("1234string",6) = "string"
Mid(<строка>, <старт> [, <длина>])	Выделяет из аргумента <строка> подстроку с указанным числом символов, начиная с позиции <старт>	Mid ("12345678",4,3) = "456"
Mid(<строка>, <старт>)	Выделяется подстрока от позиции <старт> до конца строки	Mid ("12345678",4) = "45678"
LTrim (<строка>)	Удаляет пробелы в начале строки	LTrim(" печать") = "печать"
RTrim (<строка>)	Удаляет пробелы в конце строки	RTrim("печать ") = "печать"
Trim (<строка>)	Удаляет пробелы в начале и в конце строки	Trim(" печать ") = "печать"
InStr([<старт>], <строка1>, <строка2> [, <сравнение>])	Производит поиск подстроки в строке. Возвращает позицию первого вхождения строки <строка2> в строку <строка1>, <старт> — позиция, с которой начинается поиск. Если этот аргумент пропущен, поиск начинается с начала строки	InStr("C:\Temp\test.mdb", "Test") = 9 Если искомая строка не находится в указанной строке, функция возвращает 0
InStrRev ([<старт>], <строка1>, <строка2> [, <сравнение>])	Ищет подстроку в строке, но начинает поиск с конца строки и возвращает позицию последнего вхождения подстроки. Необязательный аргумент <сравнение> определяет тип сравнения двух строк	
Replace (<строка>, <строкаПоиск>, <строкаЗамена>)	Позволяет заменить в строке одну подстроку другой. Эта функция ищет все вхождения аргумента <строкаПоиск> в аргументе <строка> и заменяет их на <строкаЗамена>	

Для сравнения строковых значений можно использовать обычные операторы сравнения числовых значений, так как при сравнении символов сравниваются их двоичные коды.

Для сравнения строковых значений также применяется оператор Like, который позволяет обнаруживать неточное совпадение, например выражение «Входной сигнал» Like «Вход\*» будет иметь значение True, так как сравниваемая строка начинается со слова

«Вход». Символ звездочка (\*) в строке заменяет произвольное число символов. Другие символы, которые обрабатываются оператором Like в сравниваемой строке:

- ? – любой символ (один);
- # – одна цифра (0–9);
- [<список>] – символ, совпадающий с одним из символов списка;
- [!<список>] – символ, не совпадающий ни с одним из символов списка.

Следующие три функции позволяют работать с массивом строк.

Split (<строка> [,<разделитель>]) – преобразует строку в массив подстрок. По умолчанию в качестве разделителя используется пробел. Данную функцию удобно использовать для разбиения предложения на слова. Однако можно указать в этой функции любой другой разделитель.

Например, Split(3, «Это тестовое предложение») возвращает массив из трех строковых значений: «Это», «тестовое», «предложение».

Join (<массивСтрок> [,<разделитель>]) – преобразует массив строк в одну строку с указанным разделителем.

Filter(<массивСтрок>,<строкаПоиск>[,<включение>] [,<сравнение>]) – просматривает массив строковых значений и ищет в нем все подстроки, совпадающие с заданной строкой.

Эта функция имеет четыре аргумента:

<строкаПоиск> – искомая строка;

<включение> – параметр (булево значение), который указывает, будут ли возвращаемые строки включать искомую подстроку или, наоборот, возвращаться будут только те строки массива, которые не содержат искомой строки в качестве подстроки;

<сравнение> – параметр, определяющий метод сравнения строк.

Еще три функции обеспечивают преобразование строк:

LCase(<строка>) – преобразует все символы строки к нижнему регистру, например функция LCase(«ПОЧТА») возвращает строку «почта»;

UCase(<строка>) – преобразует все символы строки к верхнему регистру;

StrConv(<строка>,<преобразование>) – выполняет несколько типов преобразований строки в зависимости от второго параметра. Этот параметр описывается встроенными константами, например функция StrConv(«россия»,VbProperCase) возвращает значение «Россия».

И последние две функции генерируют строки символов:

Space(<число>) – создает строку, состоящую из указанного числа пробелов;

String(<число>,<символ>) – создает строку, состоящую из указанного в первом аргументе числа символов. Сам символ указывается во втором аргументе.

*Пример 13.* Создать программу, работающую со строковыми переменными. Для этого создать форму, в метки которой выходят следующие сообщения:

1 метка: сообщается длина строки, введенной в первое текстовое поле (1 строка);

2 метка: преобразует все символы третьего текстового поля (3 строка) в заглавные буквы;

3 метка: выводит вместе содержание первого и второго текстовых полей (1 и 2 строки).

Технология выполнения

1. Откройте приложение Word, сохраните документ и перейдите в редактор VBA.

2. Создайте форму аналогично приведенному рис. 29.

3. Пропишите обработчик события кнопки ОК.



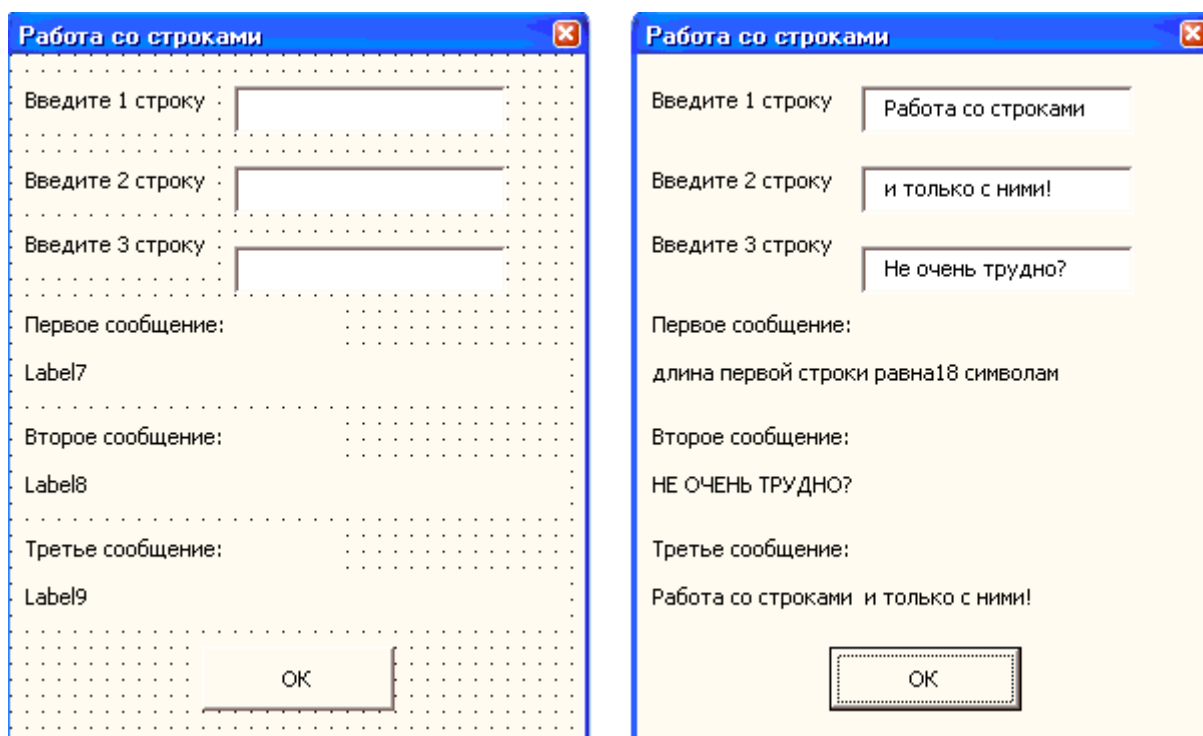


Рис. 29. Форма примера 13 в режиме конструктора и в рабочем состоянии

```
Private Sub CommandButton1_Click()
    Dim a As String
    Dim b As String
    Dim c As String
    Dim k As String
    Dim d As String
    Dim n As Integer
    a = TextBox1.Text
    n = Len(a)
    Label7.Caption = "длина первой строки равна" & n & " символом"
    c = TextBox3.Text
    k = UCase(c)
    Label8.Caption = k
    b = TextBox2.Text
    d = a + " " + b
    Label9.Caption = d
End Sub
```

4. Откомпилируйте программу.
5. Запустите форму на выполнение.

### 3.3. Процедуры и функции VBA

Описание процедур и функций VBA

Процедуры VBA бывают двух типов:

- процедуры обработки событий;
- общие процедуры.

Имя процедуры обработки события, связанного с элементом управления, состоит из имени элемента управления, символа подчеркивания и имени события, например *Закреть\_click* – процедура обработки нажатия кнопки *Закреть* в форме.

Общие процедуры VBA могут храниться в любом типе модулей VBA, так как они не связаны с конкретным объектом. Они выполняются только тогда, когда явно вызываются другими процедурами. Обычно эти процедуры реализуют какие-то общие действия, которые могут вызываться разными процедурами обработки событий.



Процедуры, как и переменные, должны быть объявлены до того, как они могут быть вызваны. Объявления общих процедур помещаются в разделе General (Общая область) модуля. Процедуры обработки событий хранятся в разделах модуля формы или отчета, соответствующих связанным с этими процедурами объектам.

В свою очередь, процедуры VBA делятся на *подпрограммы* и *функции*. Они являются фрагментами программного кода, который заключается между операторами Sub и End Sub или между Function и End Function соответственно. Процедуры-подпрограммы выполняют действия, но не возвращают значение, поэтому они не могут быть использованы в выражениях. Процедуры обработки событий представляют собой процедуры-подпрограммы. Процедуры-функции всегда возвращают значение, поэтому они обычно используются в выражениях. Общие процедуры могут быть как процедурами-подпрограммами, так и процедурами-функциями.

Синтаксис процедуры-подпрограммы VBA:

```
Sub <имяПроцедуры> (<аргумент1>, <аргумент2>, ...) <оператор1>  
<оператор2>  
End Sub
```

Список аргументов у процедуры может отсутствовать и может содержать необязательные аргументы.

Объявление каждого аргумента имеет следующий синтаксис:

```
<имяАргумента> [As <типДанных> [= <значениеПоУмолчанию>]],
```

где *<имяАргумента>* – идентификатор, составленный согласно правилам создания имен и представляющий аргумент в теле процедуры;

*<типДанных>* – это либо встроенный тип данных, либо тип, определенный пользователем. Тип данных аргумента может не указываться, и тогда считается, что он имеет тип Variant. Аргументом процедуры может быть и массив. Тогда после имени аргумента должны стоять круглые скобки.

Для необязательного аргумента может быть указано *<значение по умолчанию>*, которое будет использоваться, если этот аргумент будет опущен. Если значение по умолчанию не указано, необязательный аргумент инициализируется точно так же, как переменная, т. е. числовой аргумент – в 0, строковый – в строку нулевой длины и т. д.

Описание функции:

```
Function <имяФункции> (<аргумент1>, <аргумент2>, ...) [As  
<типЗначение>]  
<оператор1>  
<оператор2>  
<имяФункции> = <возвращаемоеЗначение>  
End Function
```

Кроме того что ключевое слово Sub заменяется на Function, в теле функции обязательно присутствует оператор присваивания имени функции какого-нибудь значения. Это значение и возвращается функцией. В заголовке функции может быть описан тип возвращаемого значения. Если этот тип не указан, функция возвращает значение Variant.

Рассмотрим два примера объявления подпрограмм и функций.

*Объявление процедуры инициализации массива*

```
Sub Init (arr() As Integer)  
Dim i As Integer, str As String  
For i = LBound(arr) To UBound(arr)  
str = "Введите количество книг на полке № " & I  
arr(i) = InputBox(str) 'функция ввода строки  
Next I  
End Sub
```

*Объявление функции, подсчитывающей сумму любого числа аргументов*

```

Function SummaVar(ParamArray varArg() As Variant) As Integer
Dim intSum As Integer, numb As Variant
For Each numb In varArg 'цикл по всем элементам массива
intSum = intSum + numb 'по умолчанию инициализируется в 0
Next numb
SummaVar = intSum 'присвоение возвращаемого значения
End Function

```

Вызов подпрограмм и функций

Чтобы использовать написанную подпрограмму или функцию, ее нужно *вызвать*.

Вызов процедуры-подпрограммы отличается от вызова процедуры-функции.

Обычно подпрограмма вызывается из другой подпрограммы или функции с помощью специального оператора VBA. Если она имеет аргументы, ей передается список *фактических параметров*.

Оператор вызова подпрограммы может использоваться в двух формах:

<имя Процедуры><список фактических параметров>

или

Call <имя Процедуры> (<список фактических параметров>).

В первом случае список фактических параметров задается без скобок, во втором – использование скобок обязательно. Но всегда список фактических параметров должен полностью соответствовать списку аргументов, заданному в объявлении подпрограммы. Все фактические параметры для обязательных аргументов должны быть перечислены в том порядке, в каком они присутствуют в описании подпрограммы, после чего могут идти параметры для необязательных аргументов.

Вызов функции имеет следующий вид:

<имя переменной>=<имя функции>(<список фактических параметров>).

*Примечание.* Список фактических параметров при вызове функции должен обязательно заключаться в кавычки.

Например, вызов объявленной в вышеприведенном примере функции SummaVar может выглядеть следующим образом:

```
Dim intShelfs (1 To 30) As Integer
```

```
Dim intS As Integer
```

```
Init intShelfs инициализируем массив intShelfs
```

```
intS = SummaVar(1,2,3,4,5,6,7,8,9)'суммируем целые числа
```

*Пример 14.* Создать программу, работающую с процедурами и функцией, параметрами которых являются значения, вводимые в текстовые поля формы. После нажатия кнопки *Счет* на форме основная программа вызывает процедуры и функцию и выводит полученные результаты в три соответствующие метки.

Процедуры и функция выполняют следующие действия:

1) первая процедура производит суммирование двух первых введенных значений а и b;

2) вторая процедура производит умножение третьего и четвертого введенных значений с и d;

3) функция вычисляет выражение  $a+b-c*d$ .

Технология выполнения

1. Откройте приложение Word, сохраните документ и перейдите в редактор VBA.

2. Создайте форму (рис. 30).

3. Пропишите обработчик события кнопки *Счет*.

При создании данной программы необходимо обратить внимание на следующие моменты. В начале программы создаются глобальные переменные, область видимости которых распространяется на все создаваемые процедуры, функцию и основной блок программы.

При создании процедур происходит считывание информации с текстовых полей. Так как вся информация, введенная в текстовые поля, априори считается текстовой, то необходимо перевести данные строковые переменные в числовые. Для чего используется функция преобразования Val. Функция работает с уже вычисленными значениями в процедурах, поэтому в теле функции вызываются процедуры, точнее их результат.

Основной блок программы также вызывает процедуры и результат созданной функции. Обратите внимание на различие написания вызова значений процедур и функции.

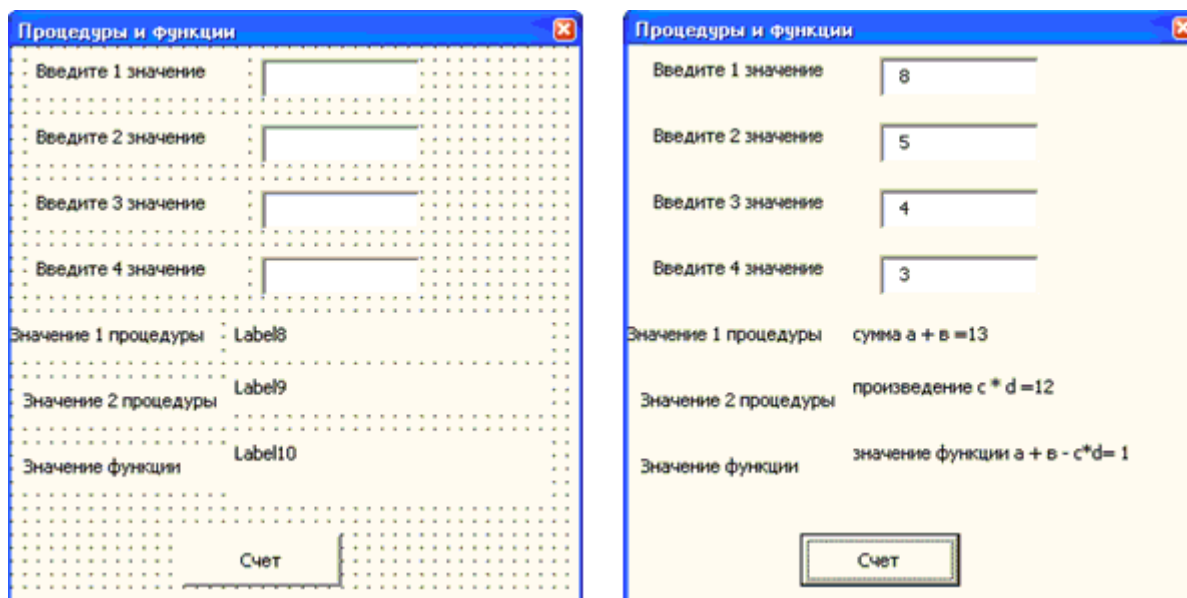


Рис. 30. Форма примера 14 в режиме конструктора и в рабочем состоянии

#### Листинг примера 14

```
Dim a, b, c, d, k, m, h As Integer
Sub summ()
a = Val(TextBox1.Text)
b = Val(TextBox2.Text)
k = a + b
End Sub
Sub umn()
c = Val(TextBox3.Text)
d = Val(TextBox4.Text)
m = c * d
End Sub
Function prim() As Integer
Call summ
Call umn
prim = k - m
End Function
Private Sub CommandButton1_Click()
Call summ
Call umn
' переменные a и b являются глобальными для запуска функции prim, где k, m являются локальными переменными в соответствующих процедурах
h = prim(a, b)
Label8.Caption = "сумма a + b =" & k
Label9.Caption = "произведение c * d =" & m
Label10.Caption = "значение функции a+b-c*d= " & h
End Sub
```

4. Откомпилируйте программу.

5. Запустите форму на выполнение.

*Пример 15.* Создать программу, которая, используя данные, считанные из текстовых полей формы, выполняет вычисление значений двух выражений:  $y = \sin(a + b) + 5$  и  $z = \operatorname{tg}(c + 3) - \cos(3 * d)$ , где  $a, b, c, d$  – значения переменных, введенных в соответствующие текстовые поля (рис. 31). Данные значения  $y$  и  $z$  вычисляются в соответствующих процедурах (процедуре) или функциях. Основной блок программы сравнивает полученные значения, и об этом выдается сообщение на форму.

работа на 5

введите число a: 7  $Y = \sin(a+b) + 5$

введите число b: 8 5,650287840157

введите число c: 9  $z = \text{tg}(c+3) - \cos(3d)$

введите число d: 5 0,1238279826

сравнение функций y и z

y > z на 5,526459857557

сравнить      закрыть

Рис. 31. Форма примера 15 в рабочем состоянии

*Примечание.* Для того чтобы определить, какое из двух чисел больше, необходимо из первого числа вычесть второе. Если разность больше нуля, то первое число больше второго (укажет, на сколько), если разность отрицательная, то второе число больше первого. При этом необходимо взять модуль от полученного значения, чтобы определить, на сколько второе число больше первого.

### Макросы

Если приходится часто использовать одни и те же команды в одной и той же последовательности, имеет смысл сохранить эту последовательность в виде VBA-программы. Такая программа называется макросом.

Запустить средство записи макросов можно командой: меню *Разработчик* + *Запись макроса*.

В результате любого из этих действий появится диалоговое окно «Запись макроса» (рис. 32).

Запись макроса

Имя макроса:  
Макрос 1

Назначить макрос

кнопке      клавишам

Макрос доступен для:  
Всех документов (Normal.dotm)

Описание:

ОК      Отмена

Рис. 32. Окно «Запись макроса»

Имена макросов должны ассоциироваться с выполняемыми ими задачами. Однако при этом следует придерживаться определенных правил.

- Имя макроса должно начинаться с буквы, а не с цифры. Имя макроса не должно содержать пробелов. Для выделения начала слов в имени макроса следует использовать прописные буквы.

- Знаки пунктуации не допускаются.

Командой для начала записи макроса будет щелчок на кнопке *OK* в диалоговом окне «Запись макроса». О том, что запись началась, можно узнать по появившейся с этого момента панели инструментов *Остановить запись*



Изменится также и указатель мыши, превратившись в небольшое изображение магнитофонной компакт-кассеты.

После начала записи необходимо выполнить обычные действия, необходимые для работы с документом или приложением, – форматирование текста, вставку рисунков, таблиц, формул и т. д. Все, что будет сделано, – и выбор команд из меню, и форматирование изображений, и печатание текста – будет сохранено в макросе.

Чтобы остановить запись после выполнения всех команд, которые нужно было записать, необходимо щелкнуть на кнопке «*Остановить запись*» в панели инструментов с тем же названием. Запись прекратится, а все записанные команды будут сохранены в виде VBA-программы.

Если необходимо выполнить команду, которая не должна быть частью макроса, щелкните на кнопке «*Пауза*» (тоже находится в панели инструментов «Остановить запись»).

Во время паузы в записи кнопка «*Пауза*» будет выглядеть нажатой. Официально теперь это будет кнопка «*Возобновить запись*». Щелкните на этой кнопке, чтобы продолжить запись в макрос необходимых действий с приложением.

Весь смысл записи макросов состоит в возможности их последующего *выполнения*. Ввиду того, что макросы являются VBA-программами, все приемы, которые используются при запуске созданных вручную VBA-программ, применимы и для автоматически записанных макросов. Всегда можно сначала открыть диалоговое окно Макрос (<Alt+F8>), выбрать в нем нужный макрос, а затем щелкнуть на кнопке *Выполнить* (рис. 33).

Независимо от сложности макроса сохраните документ перед тем, как выполнять этот макрос.

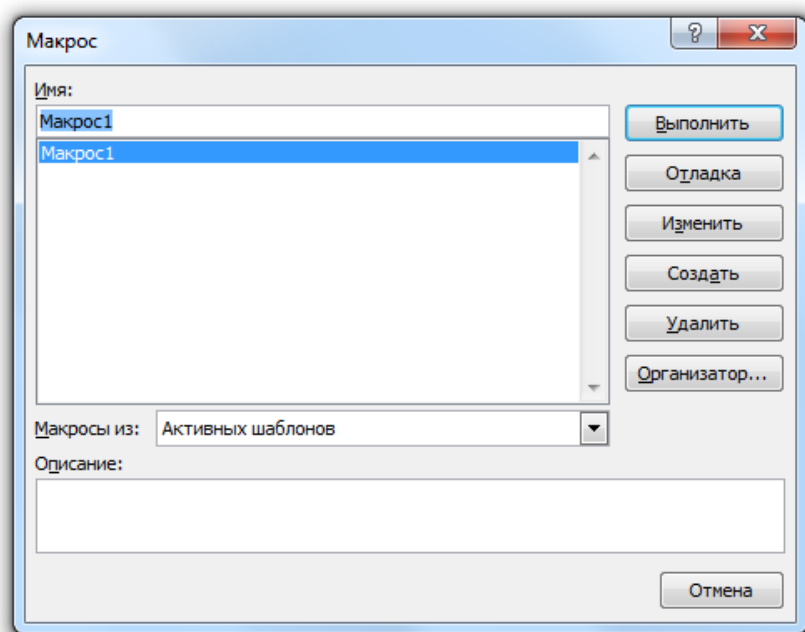


Рис. 33. Диалоговое окно «Макрос»

### Редактирование программного кода макроса в редакторе Visual Basic

После того как макрос записан, полученную VBA-программу можно отредактировать. Для этого надо:

1) выбрать команду *Разработчик + Макрос + Изменить* или нажать  $\langle \text{alt+f8} \rangle$ , чтобы открыть диалоговое окно Макрос (рис. 33);

2) выбрать нужный макрос из списка ниже поля *Имя*. Макрос, который был записан, не выбирается автоматически, поэтому, чтобы найти его, иногда придется пролистать список;

3) щелкнуть на кнопке *Изменить*. Открывается окно редактора Visual Basic с помещенным в него программным кодом, готовым для редактирования.

Задачи на закрепление материала

*Пример 16.* Создать программу, рассчитывающую значения выражений в зависимости от выбранного переключателя на форме и введенных значений в текстовые поля. Основная программа после нажатия на кнопку *Вычислить* вызывает процедуру *Shet*, рассчитывающую все необходимые выражения. После нажатия на кнопку *Заккрыть* разработанное приложение закрывается (рис. 34).

*Примечание.* При разработке данной формы используется новый элемент *Переключатель*



(optionButton), который позволяет выбрать один из нескольких взаимоисключающих параметров или действий. Наиболее часто используемые свойства элемента управления *OptionButton*:

value – возвращает True, если переключатель выбран, и False в противном случае;

enabled – допустимые значения: True (пользователь может выбрать переключатель) и False (в противном случае);

visible – допустимые значения: True (переключатель отображается во время выполнения программы) и False (в противном случае);

caption – надпись, отображаемая рядом с переключателем.

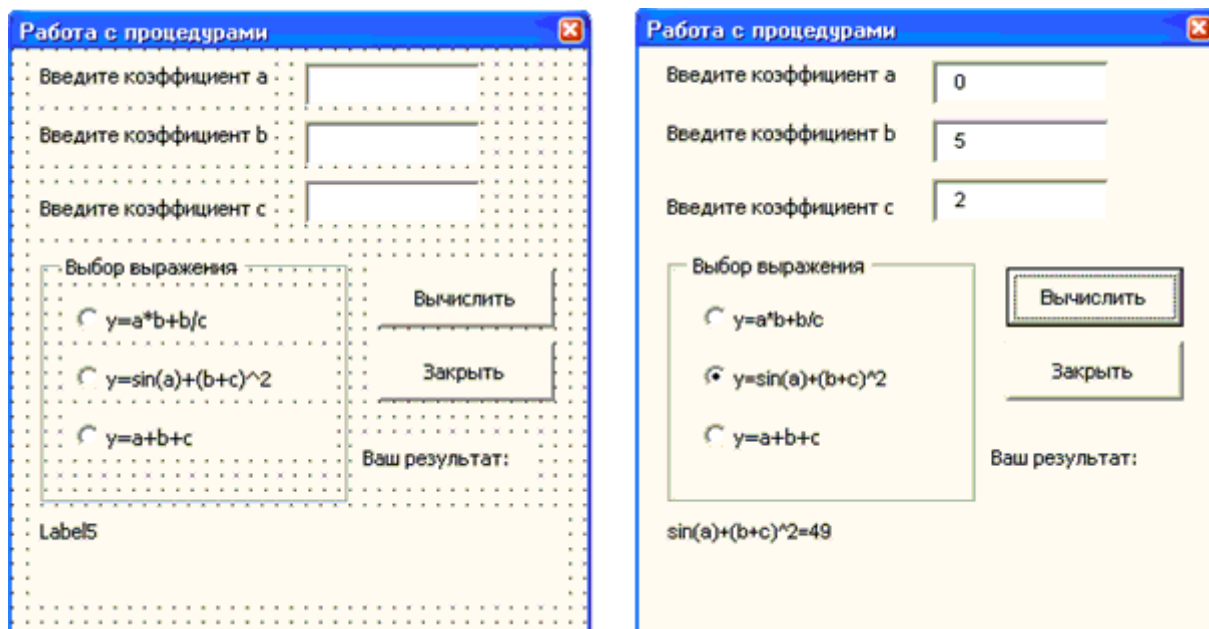


Рис. 34. Форма примера 16 в режиме конструктора и рабочем состоянии

Технология выполнения

1. Откройте приложение Word, сохраните документ и перейдите в редактор VBA.
2. Создайте форму (рис. 34).
3. Пропишите обработчики событий кнопок *Заккрыть* и *Вычислить*.

```

'Объявление глобальных переменных
Dim a, b, c, k, l, m As Double
'Создание процедуры Shet
Sub shet()
a = Val(TextBox1.Text)
b = Val(TextBox2.Text)
c = Val(TextBox3.Text)
k = a * b + b / c
l = Sin(a) + (b + c) ^ 2
m = a + b + c
End Sub
'Кнопка Вычислить
Private Sub CommandButton1_Click()
Call shet
If OptionButton1.Value = True Then
Label5.Caption = "a*b+b/c=" & k
End If
If OptionButton2.Value = True Then
Label5.Caption = "sin(a)+(b+c)^2=" & l
End If
If OptionButton3.Value = True Then
Label5.Caption = "a+b+c=" & m
End If
End Sub
'Кнопка Заккрыть
Private Sub CommandButton2_Click()
Unload Me
End Sub

```

4. Откомпилируйте программу.

5. Запустите форму на выполнение.

*Пример 17.* Создать программу, выполняющую следующее: при вводе имени пользователя, среднего балла и числа пропусков в соответствующие текстовые поля формы:

- если средний балл от 4 до 5, число пропусков меньше 100, то приложение выдает сообщение: «Имя, Вы в поощрительном списке у директора!» (рис. 35);
- если средний балл меньше 4 и число пропусков больше 100, то выдается сообщение: «Имя, Вы еще учитесь здесь?».

При другом раскладе придумайте свое сообщение.

Рис. 35. Форма примера 17 в рабочем состоянии

*Примечание.* Можно использовать различные варианты построения данной программы, один из них – использование инструкций If else или Select case.

*Пример 18.* Создать программу, выполняющую следующее: при вводе имени пользователя, среднего балла студента в текстовые поля формы и выбора числа пропусков:

- если средний балл 5, число пропусков меньше 100, то приложение выдает сообщение: «Имя, Вы молодец! У Вас меньше 100 часов пропуска и ... средний балл»;
- если средний балл от 4 до 5 и число пропусков меньше 150, то выдается сообщение: «Имя, хорошо! У Вас меньше 150 часов пропуска и ... средний балл» (рис. 36);
- если средний балл от 3 до 4 и число пропусков до 200, то выдается сообщение: «Имя, надо лучше! У Вас до 200 часов пропуска и ... средний балл»;
- если средний балл от 2 до 3 и число пропусков больше 200, то выдается сообщение: «Имя, плохо! У Вас больше 200 часов пропуска и ... средний балл».

При другом раскладе выдумайте свое сообщение.

Рис. 36. Форма примера 18 в рабочем состоянии

*Пример 19.* Создать программу, выполняющую следующее: при вводе имени пользователя и возраста в текстовые поля формы, выбора среднего балла и числа пропусков:

- если средний балл 5, число пропусков меньше 100, то приложение выдает сообщение: «Имя, Вы молодец! У Вас меньше 100 часов пропуска, средний балл равен 5, и Вам всего ... лет!»;
- если средний балл от 4 до 5, число пропусков меньше 150, то приложение выдает сообщение: «Имя, Вы молодец! У Вас меньше 150 часов пропуска, средний балл от 4 до 5, и Вам всего ... лет!» (рис. 37);

- если средний балл от 3 до 4, число пропусков меньше 200, то приложение выдает сообщение: «Имя, надо лучше! У Вас до 200 часов пропуска и средний балл от 3 до 4, и Вам уже ... лет!»;

- если средний балл от 2 до 3, число пропусков больше 200, то приложение выдает сообщение: «Имя, плохо! У Вас больше 200 часов пропуска, средний балл от 2 до 3, и Вам уже ... лет!».

*Примечание.* Предусмотреть выбор переключателей в произвольном порядке и прописать это в коде.

При другом раскладе выдумайте свое сообщение.



работа на пять

введите имя: Светлана

введите возраст: 16

Число пропусков:

- ☐ от 0 до 100
- ☒ от 100 до 150
- ☐ от 150 до 200
- ☐ больше 200

средняя оценка:

- ☐ 5
- ☒ от 4 (=) до 5 (<)
- ☐ от 3 (=) до 4 (<)
- ☐ от 2 (=) до 3 (<)

Ваш результат: Светлана, хорошо! У вас меньше 150 часов пропуска и средний балл от 4 до 5 и Вам всего 16 лет!

сообщение      закрыть

Рис. 37. Форма примера 19 в рабочем состоянии

## Лабораторная работа № 4. Создание VBA-программ

### 4.1. Элемент управления ListBox

В данной главе будут анализироваться VBA-программы, создаваемые в приложении Word без вывода результата в документ Word, поэтому программы применимы во всех приложениях пакета MS Office. Рассматриваемые примеры используют весь теоретический материал, рассмотренный в главах 1–3, а также добавляют новые возможности программирования.

Во многих создаваемых приложениях используют возможности массивов. Для доступа к элементам массивов часто применяют элемент управления ListBox. Элемент управления *ListBox* (список) создается с помощью кнопки *Список*



(ListBox). Данный элемент нужен для хранения списка значений. Из списка пользователь может выбрать одно или несколько значений, которые в последующем могут использоваться в тексте программы.

Наиболее часто используемые свойства элемента управления *ListBox*:

ListIndex – возвращает номер текущего элемента списка, нумерация элементов списка начинается с нуля;

Listcount – возвращает число элементов списка;

TopIndex – возвращает элемент списка с наибольшим номером;

columnCount – устанавливает число столбцов в списке;

textcolumn – устанавливает столбец в списке, элемент которого возвращается свойством text;

text – возвращает выбранный в списке элемент;

List (row, column) – возвращает элемент списка, стоящий на пересечении указанных строки и столбца;

RowSource – устанавливает диапазон, содержащий элементы списка;

ControlSource – устанавливает диапазон (ячейку), куда возвращается выбранный элемент из списка;

multiSelect – устанавливает способ выбора элементов списка. Допустимые значения:

- fmMultiSelectSingle – выбор только одного элемента;
- fmMultiSelectMulti – разрешен выбор нескольких элементов посредством либо щелчка, либо нажатием клавиши Пробел;
- fmMultiSelectExtended – разрешено использование клавиши shift при выборе ряда последовательных элементов списка;

selected – допустимые значения: True (если элемент списка выбран) и False (в противном случае), используется для определения выделенного текста, когда свойство multiSelect имеет значение fmMultiSelectMulti или fmMultiSelectExtended;

columnWidths – устанавливает ширину столбцов списка.

Синтаксис:

columnWidths = string,

где String – строка, устанавливающая ширину столбцов.

В примере устанавливается ширина каждого из трех столбцов списка:

With ListBox1

ColumnCount=3

ColumnWidths = “20;30;30”

end With

columnHeads – допустимые значения: True (выводятся заголовки столбцов раскрывающего списка) и False (в противном случае);

listStyle – допустимые значения:

- fmListStylePlain – выбранный элемент из списка выделяется цветом;

- `fmListStyleOption` – перед каждым элементом в списке располагается флажок, и выбор элемента из списка соответствует установке этого флажка;
  - `boundColumn` – устанавливает тип, возвращаемый свойством `Value`. А именно:
    - если свойство `boundColumn` равно 0, то свойство `value` возвращает индекс выбранной строки;
    - если свойство `boundColumn` принимает значение из диапазона от 1 до количества столбцов в списке, то свойство `value` возвращает элемент из выбранной строки, стоящей в столбце, определенном свойством `boundcolumn`.
- Наиболее часто используемые методы элемента управления *ListBox*:
- `clear` – удаляет все элементы из списка;
  - `RemoveItem (index)` – удаляет из списка элемент с указанным номером, где `index` – номер элемента;
  - `AddItem ([item[,varIndex]])` – добавляет элемент в список, где `item` – элемент (строковое выражение), добавляемый в список, и `varIndex` – номер добавляемого элемента.
- Заполнить список можно одним из следующих способов.

Поэлементно, если список состоит из одной колонки

```
With ListBox1
```

```
AddItem «июнь»
```

```
AddItem «июль»
```

```
AddItem «август»
```

```
ListIndex=0
```

```
End With
```

Массивом, если список состоит из одной колонки

```
With ListBox1
```

```
List=Array(«июнь», "июль", "август")
```

```
ListIndex = 1
```

```
End With
```

Из диапазона A1:B4, в который предварительно введены элементы списка. Результат выбора (индекс выбранной строки) выводится в ячейку C1

```
With ListBox1
```

```
ColumnCount = 2
```

```
RowSource = «A1:B4»
```

```
ControlSource = «C1»
```

```
BoundColumn=0
```

```
End With
```

Поэлементно, если список состоит из нескольких колонок, например двух

```
With ListBox1
```

```
ColumnCount = 2
```

```
AddItem «июнь»
```

```
List(0,1)="сессия"
```

```
AddItem «июль»
```

```
List(1,1)="каникулы"
```

```
AddItem «август»
```

```
List(2,1)="отработка"
```

```
End With
```

Массивом, если список

состоит из нескольких

колонок, например двух

```
Dim A(2,1) As string
```

```
A(0,0) = «июнь»
```

```
A(0,1) = «сессия»
```

```
A(1,0) = «июль»  
A(1,1) = «каникулы»  
A(2,0) = «август»  
A(2,1) = «отработка»  
With ListBox1  
ColumnCount = 2  
List = A  
End With
```

Задачи на закрепление материала

*Пример 20.* Создать программу, которая при вводе имени пользователя и числа от 1 до 10 в текстовые поля формы выдает в метку label1 предсказание в зависимости от введенного значения.

При разработке программы использовать одномерный массив, объявляемый в процедуре *Fortuna*, содержащей все возможные предсказания.

Технология выполнения

1. Откройте приложение Word, сохраните документ и перейдите в редактор VBA.
2. Создайте форму (рис. 38).
3. Пропишите обработчики событий нажатия на кнопки «Вывести предсказание» и «Заккрыть».

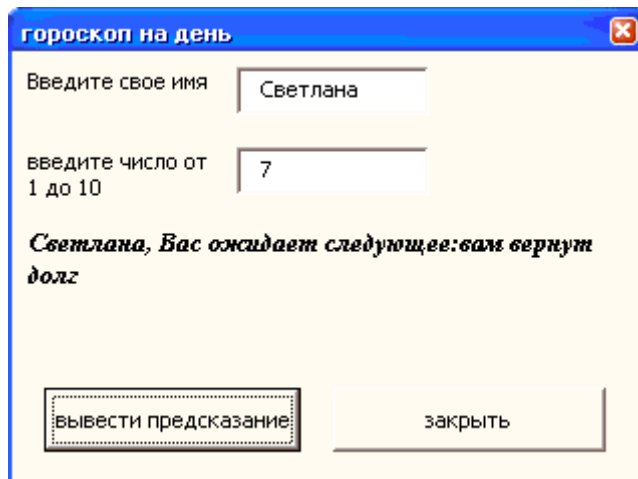


Рис. 38. Форма примера 20 в рабочем состоянии

При обработке процедуры нажатия кнопки *Вывести* можно опираться на нижеприведенный листинг.

*Примечание.* Обратите внимание на создание процедуры, содержащей массив предсказаний, и вызов этой процедуры из процедуры нажатия кнопки «Вывести предсказание», учитывая «защиту от дурака».

```

Sub fortune(a As String, b As Integer)
Dim today(1 To 10)
today(1) = "Вы станете богатым и знаменитым за 15 минут"
today(2) = "Вам предстоит обед с незнакомцем"
today(3) = "Стоимость Ваших вкладов удвоится!"
today(4) = "Вы получите большой букет от своего почитателя"
today(5) = "Вы опоздаете на пару"
today(6) = "Все Ваши мечты сбудутся"
today(7) = "Вам вернут долг"
today(8) = "Вы выучите лекцию и ответите на опрос"
today(9) = "Вы встретите своего давнего знакомого"
today(10) = "На Вас обратят внимание"
Label3.Caption = a & ", Вас ожидает следующее:" & today(b)
End Sub
Private Sub CommandButton1_Click()
Dim a As String
Dim b As Integer
a = TextBox1.Text
b = Val(TextBox2.Text)
If b > 0 And b <= 10 Then
Call fortune(a, b)
Else: Label3.Caption = "Вы ввели не то число!"
End If
End Sub

```

4. Откомпилируйте программу.
5. Запустите приложение на выполнение.

*Пример 21.* Создать форму, в текстовые поля которой вводятся имя и пароль. Если пароль введен правильно, то пользователь получает доступ к списку некоторой группы, отражаемой в элементе listbox, при этом выдается сообщение о допуске пользователя к списку. Список организуется программно.

Технология выполнения

1. Откройте приложение Word, сохраните документ и перейдите в редактор VBA.
2. Создайте форму (рис. 39).

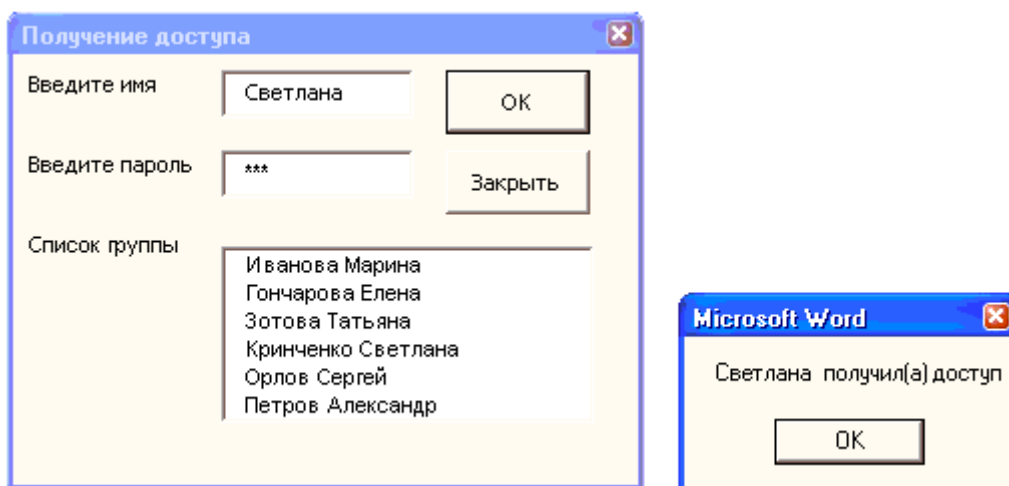


Рис. 39. Форма примера 21 в рабочем состоянии. Сообщение о допуске

3. Переименуйте форму с Name на frmOne. Для этого активизируйте форму (щелкните на ней), перейдите в окно свойств объекта (properties), выделите свойство Name (первая строка) и пропишите frmOne.

4. Создайте текстовые поля textbox1 и textbox2. Первое будет использоваться для ввода имени, второе – для ввода пароля, поэтому переименуйте их на txtName и txtPassword (аналогично пункту 3).

5. Как правило, вводимый пароль не отображается в поле. Чтобы вводимые символы пароля заменялись звездочками, выделите поле пароля, затем в окне *Properties* найдите свойство *PasswordChar* и задайте для него значение «\*».

6. Добавьте кнопки. Переименуйте первую кнопку на *btnOK*. Кнопка ОК должна быть кнопкой по умолчанию. Это означает, что на нажатие клавиши Enter форма должна реагировать так же, как и на щелчок этой кнопки. Для этого измените значение свойства *Default* кнопки *btnOK* на *True*.

7. Переименуйте вторую кнопку на *btnCancel*. Щелчок кнопки *Cancel* обычно эквивалентен нажатию клавиши *Escape*. Чтобы в вашем окне было именно так, задайте значение «True» для свойства *Cancel*.

8. Создайте процедуру закрытия формы.

9. Создайте процедуру работы кнопки ОК. В заготовке процедуры

```
Private Sub btnOK_Click()
```

```
End Sub
```

вставьте оператор *MsgBox Txtname & «получил(а) доступ»*. В этом операторе текст, введенный в поле *txtName*, объединяется с поясняющей строкой и выводится в информационном окне.

10. Создайте процедуру для проверки пароля. Закроем доступ к кнопке ОК, пока не введен верный пароль. Для этого измените свойство *Enabled* кнопки ОК на *False* (доступ к кнопке закрыт) и введите новую процедуру:

```
Private Sub CheckOK()
```

```
Const pas = «abc» 'Пароль
```

```
If Txtpassword = pas Then 'Если текст в поле
```

```
Txtpassword совпадает с паролем, то btnOK.Enabled = True 'доступ к кнопке ОК  
открыт
```

```
Else 'иначе
```

```
btnOK.Enabled = False 'доступ к кнопке ОК закрыт
```

```
End If
```

```
End Sub
```

Процедура должна запускаться при вводе нового пароля. Для этого в окне формы щелкните дважды второе текстовое поле (предназначенное для пароля). В появившуюся процедуру

```
Private Sub Txtpassword_Change()
```

```
End Sub
```

введите единственную команду *CheckOK*, которая при внесении любых изменений в поле пароля будет запускать процедуру проверки пароля.

11. Заполните список группы. Расположите на форме элемент управления *ListBox*. По умолчанию его имя *ListBox1*. Оставим его в таком виде. Заполнить список можно в операторе *With*:

```
With ListBox1
```

```
List = Array(«Иванова Марина», «Гончарова Елена», «Зотова Татьяна», «Кринченко  
Светлана», «Орлов Сергей», «Петров Александр»)
```

```
End With
```

Здесь *List* – свойство элемента *ListBox*, которое является массивом. Вставьте этот оператор в уже имеющуюся процедуру *btnOK\_Click*.

12. Проверка работы формы. После запуска формы введите свое имя и верный пароль. Должна стать доступной кнопка ок. После ее нажатия появляются сообщение «Имя получил(а) доступ» и заполненный список. После нажатия кнопки *Закрыть* форма должна закрыться.

Дополнительные задания

13. Дополните список с помощью метода *AddItem*. Синтаксис:

ИмяСписка. *AddItem* «новый элемент списка».

14. Обработайте событие, связанное со щелчком по форме (мимо элементов управления). Пусть в ответ на это выводится какое-нибудь сообщение типа «Вы не попали по кнопке!».

#### 4.2. Элементы управления *ComboBox*, *OptionButton* и *Frame*

Поле со списком

Элемент управления *ComboBox*



(поле со списком) создается с помощью соответствующего элемента. Элемент управления *ComboBox* применяется для хранения списка значений. Он сочетает в себе функциональные возможности списка *ListBox* и поля *TextBox*. В отличие от *ListBox*, в элементе управления *ComboBox* отображается только один элемент списка. Кроме того, у него отсутствует режим выделения нескольких элементов списка, но он позволяет вводить значение, используя поле ввода, как это делает элемент управления *TextBox*.

Свойства объекта *ComboBox*, такие как *ListIndex*, *ListCount*, *Enabled*, *List*, и методы *Clear*, *RemoveItem* и *AddItem* аналогичны соответствующим свойствам и методам списка *ListBox*. Кроме того, у него есть ряд уникальных свойств:

*DropDownStyle* – устанавливает вид раскрывающегося списка. Допустимые значения:

- *fmDropDownStylePlain* – кнопка без символов;
- *fmDropDownStyleArrowDisplays* – кнопка со стрелкой;
- *fmDropDownStyleEllips* – кнопка с эллипсом;
- *fmDropDownStyleReduce* – кнопка с линией.

*ListRows* – устанавливает число элементов, отображаемых в раскрываемом списке.

*MatchRequired* – допустимые значения: *True* (нельзя ввести значения) и *False* (в противном случае).

*MatchFound* – допустимые значения: *True* (среди элементов раскрывающегося списка имеется элемент, совпадающий с вводимым в поле ввода раскрывающегося списка) и *False* (в противном случае).

Переключатель и рамка

Элемент управления *OptionButton*



(переключатель) создается с помощью соответствующего элемента. Он позволяет выбрать один из нескольких взаимоисключающих параметров или действий. Переключатели обычно отображаются группами, обеспечивая возможность выбора альтернативного варианта (см. п. 3.3).

Элемент управления *Frame*



(рамка) создается с помощью соответствующего элемента. Он используется для визуальной группировки элементов управления. Основным свойством рамки является *caption*, отображающее надпись рамки.

Задачи на закрепление материала

*Пример 22.[1]* Создать программу, которая позволяет при выделении из списка формы нескольких чисел производить суммирование, находить произведение или среднее значение в зависимости от выбора действия (операции).

Нажатие кнопки «Вычислить» должно привести к выполнению выбранной операции над выбранными числами и выводу результата в поле «Результат» (рис. 40).

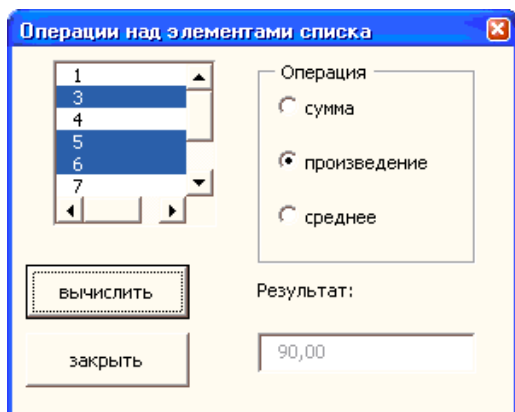


Рис. 40. Разработанная форма примера 22 в рабочем состоянии

Технология выполнения

Обсудим, как приведенная ниже программа решает перечисленные пункты задачи и что происходит в программе.

UserForm\_Initialize

1. Активизирует диалоговое окно

2. Запрещает ввод данных в поле *Результат*.

3. Связывает с кнопками *Вычислить* и *Закреть*, а также с переключателями всплывающие подсказки

Нажатие кнопки *Вычислить* запускает на выполнение процедуру *CommandButton1\_Click*

Определяет, какой переключатель выбран. В зависимости от выбранного переключателя производит действие над выбранными в списке числами. Найденное число выводится в поле *Результат*

Нажатие кнопки *Закреть* запускает на выполнение процедуру *CommandButton2\_Click*

Закрывает диалоговое окно

Процедура нажатия кнопки «*Вычислить*» производит вычисления с элементами списка в зависимости от выбранной операции. Вводятся переменные  $n$  и  $i$ , где  $n$  играет роль счетчика числа выбранных элементов из списка, а  $i$  – вспомогательная переменная. Также вводятся переменные, которые будут содержать результат вычислений: *сумма*, *произведение* и *среднее*. Все полученные вычисления присваиваются переменной *Результат*, которая и выдается в текстовое поле.

*Процедура нажатия кнопки Вычислить*



```

Private Sub CommandButton1_Click()
Dim i As Integer
Dim n As Integer
Dim Сумма As Double
Dim Произведение As Double
Dim Среднее As Double
Dim Результат As Double
' При выборе первого переключателя вычисляется сумма
If OptionButton1.Value = True Then
Сумма = 0
With ListBox1
For i = 0 To .ListCount - 1
If .Selected(i) = True Then
Сумма = Сумма + .List(i)
End If
Next i
End With
Результат = Сумма
End If
' При выборе второго переключателя вычисляется произведение выбранных элементов
If OptionButton2.Value = True Then
Произведение = 1
With ListBox1
For i = 0 To .ListCount - 1
If .Selected(i) = True Then
Произведение = Произведение * .List(i)
End If
Next i
End With
Результат = Произведение
End If
' При выборе третьего переключателя вычисляется среднее арифметическое
If OptionButton3.Value = True Then
Среднее = 0
n = 0
With ListBox1
For i = 0 To .ListCount - 1
If .Selected(i) = True Then
n = n + 1
Среднее = Среднее + .List(i)
End If
Next i
End With
Результат = Среднее / n
End If
' Полученное значение Результат выводится в текстовое поле
TextBox1.Text = CStr(Format(Результат, "Fixed"))
End Sub
'Процедура нажатия кнопки "Заккрыть"
Private Sub CommandButton2_Click()
UserForm1.Hide
End Sub

```

Процедура инициализации диалогового окна заключается в заполнении списка и выводе его на форме, организации всплывающих подсказок, запрещении ввода текста в поле результата на форме.

```

Private Sub UserForm_Initialize()
With ListBox1
List = Array(1, 3, 4, 5, 6, 7, 8, 10)
ListIndex = 0 'начальная индексация массива
MultiSelect=fmMultiSelectMulti
End With

```

```

' Первоначальный выбор переключателя Сумма при инициализации диалогового
окна и задание текста всплывающих подсказок у переключателей With OptionButton1.Value
= True
    ControlTipText ="Сумма выбранных элементов"
End With
OptionButton2.ControlTipText ="Произведение выбранных элементов"
OptionButton3.ControlTipText = «Среднее значение выбранных элементов»
' Поле Результат не доступно для пользователя
TextBox1.Enabled = False
' Назначение клавише <Enter> функции кнопки Вычислить и задание текста
всплывающей подсказки
With CommandButton1
    Default = True
    ControlTipText = «Нахождение результата»
End With
' Назначение клавише <Esc> функции кнопки Заккрыть и задание текста
всплывающей подсказки
CommandButton2.Cancel = True
' Задание заголовка пользовательской формы
UserForm1.Caption = «Операции над элементами списка»
UserForm1.Show
End Sub

```

*Примечание.* Интересной особенностью приводимой процедуры инициализации UserForm\_initialize является то, что заголовок диалогового окна вводится программно при помощи свойства Caption, а не вручную при помощи окна Properties.

*Пример 23.* Разработать программу нахождения среднего балла студентов, выбранных из списка в диалоговом окне «Средний балл». Список содержит фамилию студента и его средний балл (двумерный массив) (рис. 41).

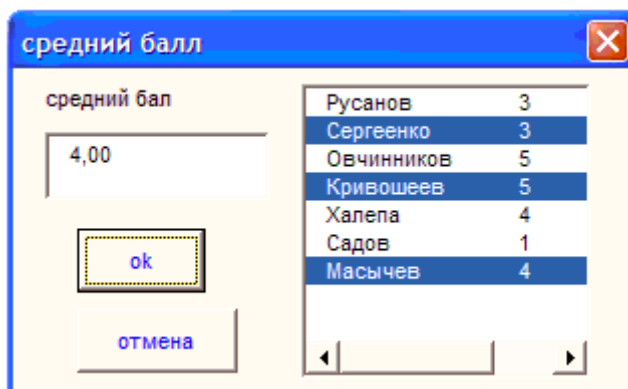


Рис. 41. Разработанная форма примера 23 в рабочем состоянии

Дополнительно для каждого элемента управления следует прописать процедуры для осуществления всплывающих подсказок.

При выполнении задания опирайтесь на приводимые ниже примечания.

*Примечание.* Обратите внимание на то, что в отличие от примера 22 данный пример требует ввода двумерного списка: столбца фамилий и столбца оценок. Фактически разница заключается лишь в том, что в приведенном примере массив ListBox1.List, отвечающий за список элементов, был одномерным, а в данном случае должен быть двумерным. Сделать это можно двумя способами:

- непосредственно заполнить двумерный массив ListBox1.List;
- задать произвольный двумерный массив, а затем присвоить его значение для ListBox1.List (см. п. 3.1 и 4.1).

Соответственно, для расчета среднего балла следует воспользоваться только элементами второго столбца массива, т. е. вторая координата массива `ListBox1.List` должна быть фиксирована на 1 (если индексация массива положена с нуля) или на 2 (если индексация массива положена с 1).

### 4.3. Элементы управления `MultiPage`, `ScrollBar`, `SpinButton`[2]

#### Коллекция `Controls`

Для доступа к набору элементов управления диалогового окна можно использовать коллекцию `Controls`, включающую все элементы управления окна. Каждый элемент управления имеет в этой коллекции индекс, значение которого может быть числом или строкой. Для первого элемента управления индекс равен 0. Числовые индексы определяются порядком размещения элементов в коллекции. Строковое значение индекса соответствует имени (`Name`) элемента.

*Пример 24.* Создать форму, имеющую пять элементов: метка, текстовое поле, список, две кнопки. Ввести в текстовое поле и список по умолчанию текст «поле 1», «список 1» (свойство `text` соответственно у каждого элемента). Первая кнопка «Нажми» выполняет следующее действие: вызывается диалоговое окно, в котором запрашивается разрешение на удаление очередного элемента формы (их пять). Ответ «да» или «нет» выполняет соответствующее действие. Кнопка «Заккрыть» закрывает диалоговое окно.

*Примечание.* Используя коллекцию `Controls`, программно в цикле организовать скрытие (не удаление!) элементов управления диалогового окна `MyForm` (свойство `Name` формы).

Технология выполнения

1. Запустите приложение `Word`, сохраните новый документ.
2. Создайте форму в режиме конструктора (рис. 42).

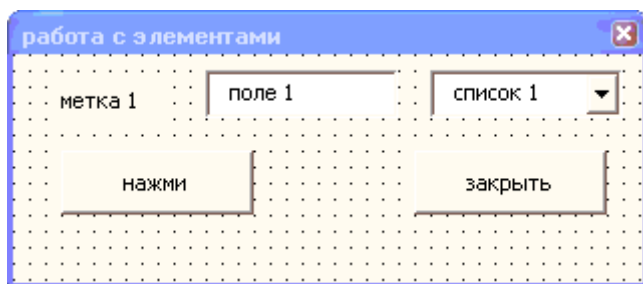


Рис. 42. Форма примера 24

3. Обработайте кнопки *Нажми* и *Заккрыть*.

```
'Кнопка Нажми
Private Sub CommandButton1_Click()
For Each Ctrl In myForm.Controls 'цикл по всем элементам управления
msgCode = vbYesNo + vbQuestion
'Вопрос об очередном элементе управления:
Answer = MsgBox(prompt:="Скрыть элемент " & Ctrl.Name, Buttons:=msgCode, Title:="Вопрос")
If Answer = vbYes Then 'ответ "Да"
Ctrl.Visible = False 'скрыть очередной элемент
End If
Next Ctrl
End Sub

Кнопка Заккрыть
Private Sub CommandButton2_Click()
Unload Me
End Sub
```

4. Откомпилируйте приложение.
5. Запустите на выполнение (рис. 43, 44, 45).

*Примечание.* Скрытие ненужных элементов в форме можно произвести и другим способом (например, если в форме слишком много элементов, то до нужного добираться по циклу не рационально). Для этого используют свойство `Click` элементов управления.

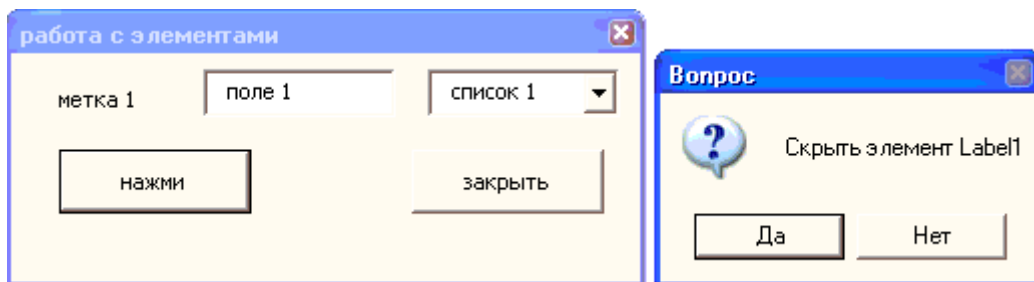


Рис. 43. Первоначальный запуск формы и вызов процедуры кнопки *Нажми*

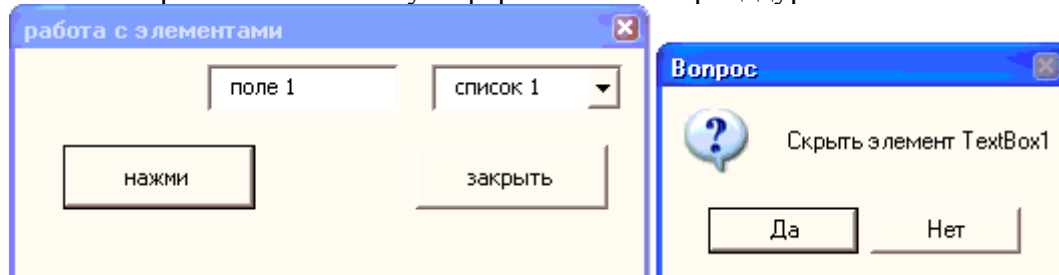


Рис. 44. Выполнение скрывтия элемента label1 и следующий запрос

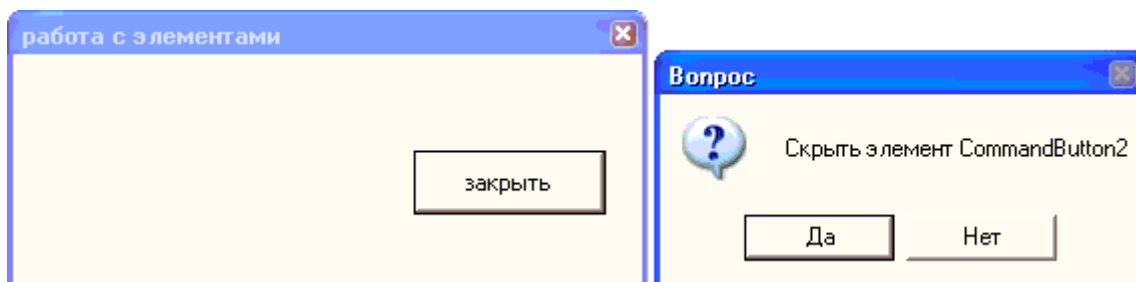


Рис. 45. Вид формы перед удалением последнего элемента управления

В этом случае необходимо только щелкнуть по нужному элементу и прописать код, для каждого элемента свой:

```
Private Sub Label1_Click()
    msgCode = vbYesNo + vbQuestion
    Answer = MsgBox(prompt:="Скрыть элемент", Buttons:=msgCode, Title:="Вопрос")
    If Answer = vbYes Then
        Label1.Visible = False
    Else
        Label1.Visible = True
    End If
End Sub
```

В результате работа формы будет более рациональной (рис. 46).

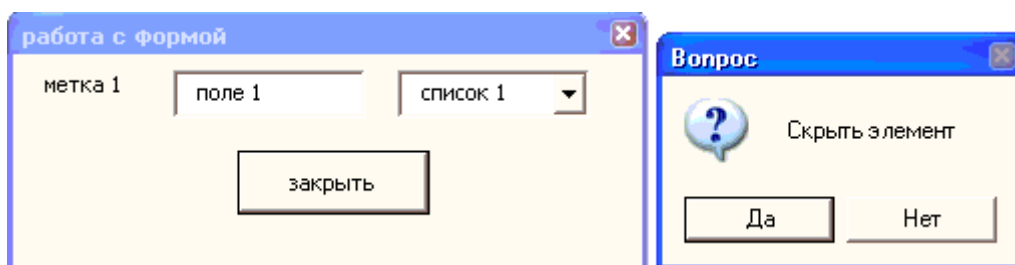


Рис. 46. Работа формы по щелчку скрываемого элемента

### Набор страниц MultiPage

Элемент управления MultiPage объединяет несколько независимых диалоговых окон – страниц (вкладок). Заголовки страниц обычно видны на одной из сторон элемента на их закладках, а переход на страницу происходит после щелчка по ее закладке. Этот простой

переход с одной страницы на другую и делает MultiPage удобным средством для представления разнородных данных, относящихся к одному объекту. Такие данные в «бумажных» офисах хранятся обычно в отдельных папках и образуют дела, досье и т. д. Каждая страница из Multipage – это объект типа Page, а все они включены в коллекцию Pages (страницы). При создании элемента MultiPage в него автоматически помещаются две страницы с именами Page1 и Page2. Имена можно изменять, присутствует возможность добавления и новых страниц. Рассмотрим основные свойства набора страниц.

- Свойство count определяет, какое количество страниц возвращается.
- Свойство value для элемента multipage определяет номер текущей активной страницы в коллекции pages.
- Свойство selecteditem (его можно только читать) возвращает текущую активную страницу (как объект). Его можно использовать для считывания и установки свойств этой страницы и входящих в нее элементов управления.
- Свойство style определяет, в каком виде представляются заголовки страниц. По умолчанию оно равно fmTabstyletabs = 0 и задает представление заголовков в виде закладок в полосе заголовков. Каждая закладка с заголовком находится внутри границ своей страницы. Если значение fmtabstylebuttons = 1, то заголовок каждой страницы находится на отдельной кнопке, расположенной в полосе заголовков. Переход на страницу происходит после выбора кнопки с ее заголовком. Если же значение fmtabstylenone = 2, то полоса с заголовками страниц на экран не выводится.
- Свойство taborientation задает расположение полосы с заголовками страниц (табл. 13).

Таблица 13 -Значения свойства TabOrientation

Константа	Значение	Расположение заголовков
fmTabOrientationTop	0	Наверху
fmTabOrientationBottom	1	Внизу
fmTabOrientationLeft	2	На левой стороне
fmTabOrientationRight	3	На правой стороне

- Булево свойство multirow позволяет создать несколько полос с закладками (по умолчанию его значение равно false, что соответствует одной полосе закладок).
- Свойства tabfixedheight и tabfixedwidth устанавливают или возвращают высоту и ширину закладки (в точках). При значении 0 ширина закладок устанавливается автоматически, так чтобы в каждой закладке помещалось ее название и занимало всю ширину элемента. При значениях больше 0 у всех закладок одинаковые размеры, заданные свойством TabFixedWidth. Минимально возможный размер закладки – 4 точки.

Для того чтобы редактировать свойства элемента MultiPage, необходимо выделить этот элемент так, чтобы рамка вокруг элемента выделилась точками (рис. 47). Для этого щелкните по самой рамке, если она выделена черной штриховкой.

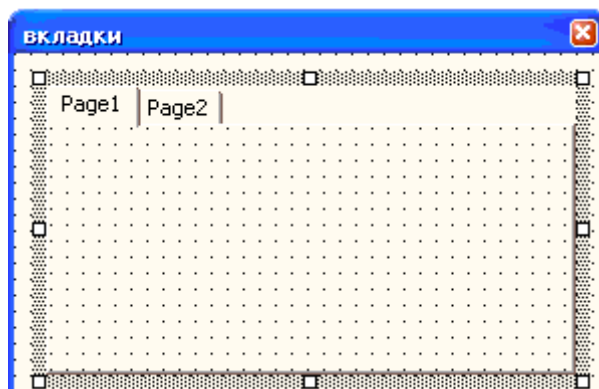


Рис. 47. Рамка элемента multipage

Если необходимо установить свойства самих вкладок, необходимо щелкнуть по ним, после чего рамка станет выделяться черной штриховкой (рис. 48).

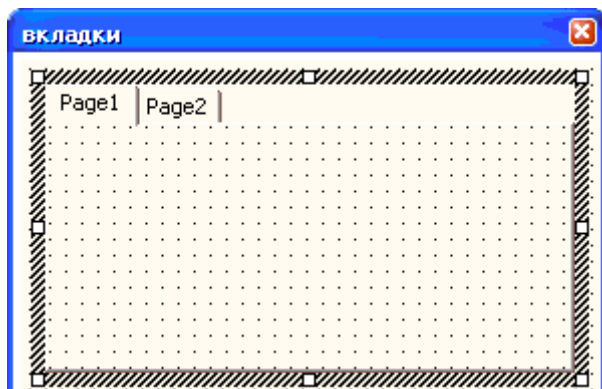


Рис. 48. Рамка элемента page

Если необходимо добавить еще одну вкладку в режиме конструктора, то необходимо щелкнуть правой кнопкой мыши по последней вкладке (странице) и выбрать команду New Page (рис. 49).

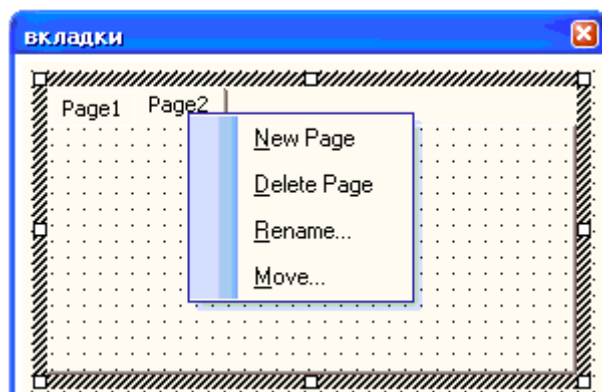


Рис. 49. Добавление новой страницы

Если необходимо переименовать страницу, ввести всплывающие подсказки и установить номер активной по умолчанию страницы, необходимо выбрать команду Rename (рис. 49), в результате которой появляется диалоговое окно, где устанавливаются перечисленные свойства (рис. 50 и 51).

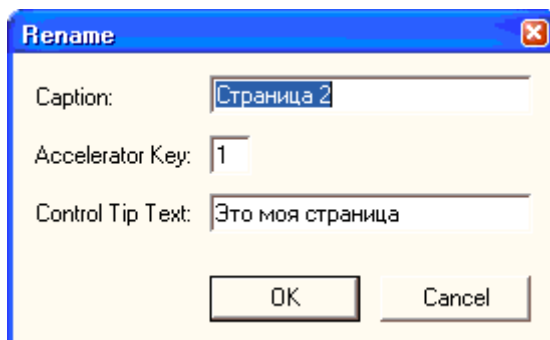


Рис. 50. Установление свойств вкладки

*Пример 25.* Создать форму, включающую в себя ряд вкладок (страниц), содержащих вопросы некоторого теста. На каждой странице находятся очередной вопрос и кнопка

«Ответ», фиксирующая выбор правильного (неправильного) ответа. После перехода на следующую страницу (щелчок по корешку страницы с соответствующим именем) действия повторяются. Итоговая страница содержит кнопку «Результат», которая выводит правильность ответов и сумму набранных баллов (1 правильный ответ = 1 балл).

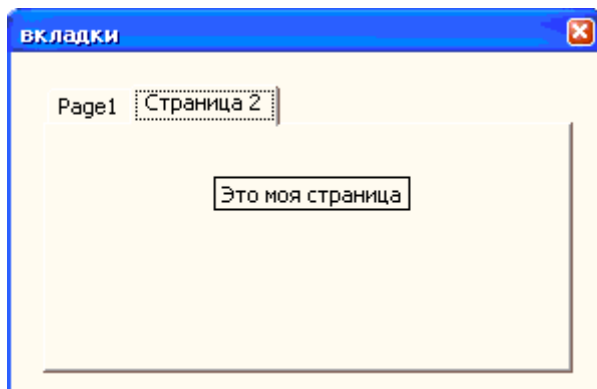


Рис. 51. Свойства в действии

Создать программу, выполняющую все требования примера.

Технология выполнения

1. Запустите приложение Word и сохраните документ.
2. Перейдите в редактор VBA и создайте форму, содержащую 4 страницы (рис. 52).

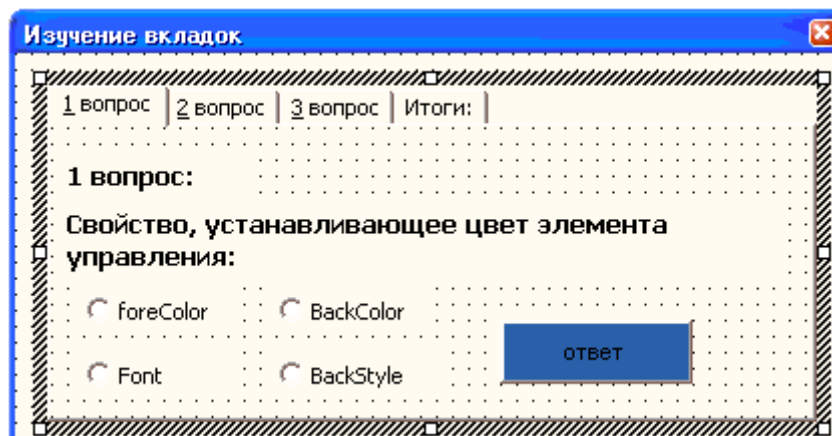


Рис. 52. Первая страница теста

3. Оформите первую страницу (рис. 52) и обработайте кнопку «Ответ».

```
Dim a As Integer
Private Sub CommandButton1_Click()
If OptionButton3.Value = True Then
a = a + 1
Label11.Caption = "Ответ верен"
Else
Label11.Caption = "Ответ неверен"
End If
End Sub
```

4. Оформите вторую страницу теста (рис. 53) и обработайте кнопку «Ответ».



```

Private Sub CommandButton2_Click()
If OptionButton5.Value = True Then
a = a + 1
Label12.Caption = "Ответ верен"
Else
Label12.Caption = "Ответ неверен"
End If
End Sub

```

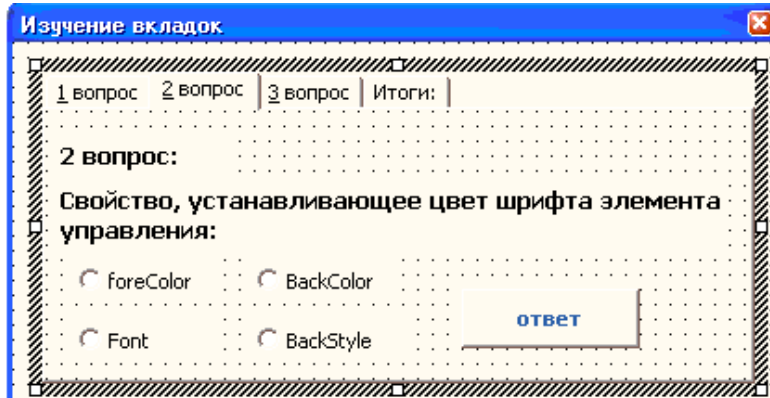


Рис. 53. Вторая страница теста

5. Оформите третью страницу теста (рис. 54) и обработайте кнопку «Ответ».

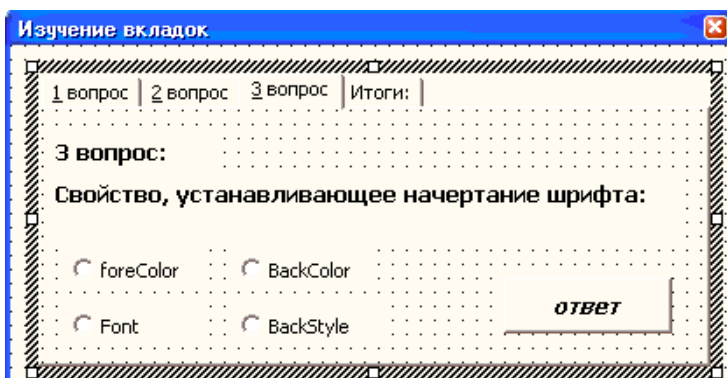


Рис. 54. Третья страница теста

```

Private Sub CommandButton3_Click()
If OptionButton10.Value = True Then
a = a + 1
Label13.Caption = "Ответ верен"
Else
Label13.Caption = "Ответ неверен"
End If
End Sub

```

6. Оформите четвертую страницу теста (рис. 55) и обработайте кнопку «Результат».

```

Private Sub CommandButton4_Click()
Label15.Caption = "вы набрали " & a & «балл(a)»
End Sub

```



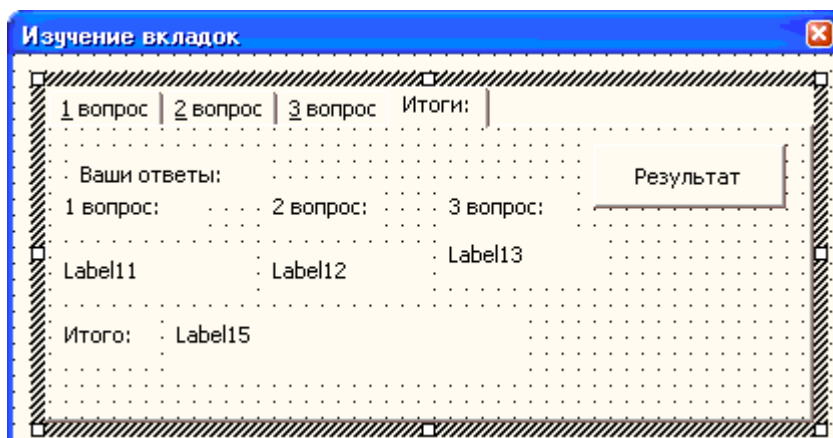


Рис. 55. Итоговая страница теста

*Примечание.* Напоминаем, что все предлагаемые листинги примерные и их можно и нужно редактировать по своему усмотрению.

7. Откомпилируйте программу и запустите на выполнение.

### Полоса прокрутки ScrollBar

Элемент управления ScrollBar представляет собой вертикальную или горизонтальную полосу, по краям которой расположены кнопки прокрутки, а внутри перемещается бегунок. Значение Value, устанавливаемое в полосе прокрутки или возвращаемое ей, – число, определяемое положением бегунка и границами, определенными в свойствах Min и Max. Рекомендуемые значения этих границ – от –32 767 до + 32 767 (по умолчанию установлен диапазон [0, 32 767]).

Обычно полоса прокрутки используется в паре с другим элементом управления (textbox), в котором она может отображать или от которого может получать свое значение.

Свойства полосы прокрутки следующие.

- Горизонтальная или вертикальная ориентация полосы прокрутки определяется свойством orientation. При его значении по умолчанию fmorientationauto = –1, ориентация полосы определяется автоматически в зависимости от ее размера по горизонтали и вертикали (большой размер задает ориентацию). Значение fmorientationvertical = 0 задает вертикальную ориентацию полосы, fmorientationhorizontal = 1 – горизонтальную.

- Свойства largechange и smallchange определяют, на сколько изменится значение value при одном щелчке поверхности полосы между кнопкой прокрутки и бегунком в первом случае и при щелчке кнопки прокрутки – во втором. Эти же свойства указывают, на сколько при этом смещается бегунок. По умолчанию оба свойства равны 1. Рекомендуемая область значений обоих свойств – от –32,767 до 32,767.

- Свойство delay (задержка) определяет в миллисекундах время, через которое последовательно возникают события change, если пользователь непрерывно щелкает кнопку прокрутки или левую кнопку мыши, указывающей на полосу прокрутки. По умолчанию устанавливается значение в 50 миллисекунд.

- Свойство proportionalthumb определяет размер бегунка: true – размер бегунка пропорционален размеру области прокрутки (это значение по умолчанию); false – система определяет фиксированный размер бегунка.

*Пример 26.* Создать форму, содержащую два элемента scrollbar, два текстовых поля и кнопку, выводящую результат вычисления в метку на форме. Программно отражать значения в текстовых полях формы при движении бегунка. После нажатия на кнопке «Вычислить» программа считывает значения с текстовых полей, переводит эти значения в числовые и суммирует. Результат отражается в соответствующей метке на форме.

Технология выполнения

1. Активизируйте приложение Word и сохраните документ.
2. Перейдите в редактор VBA и создайте форму (рис. 56).

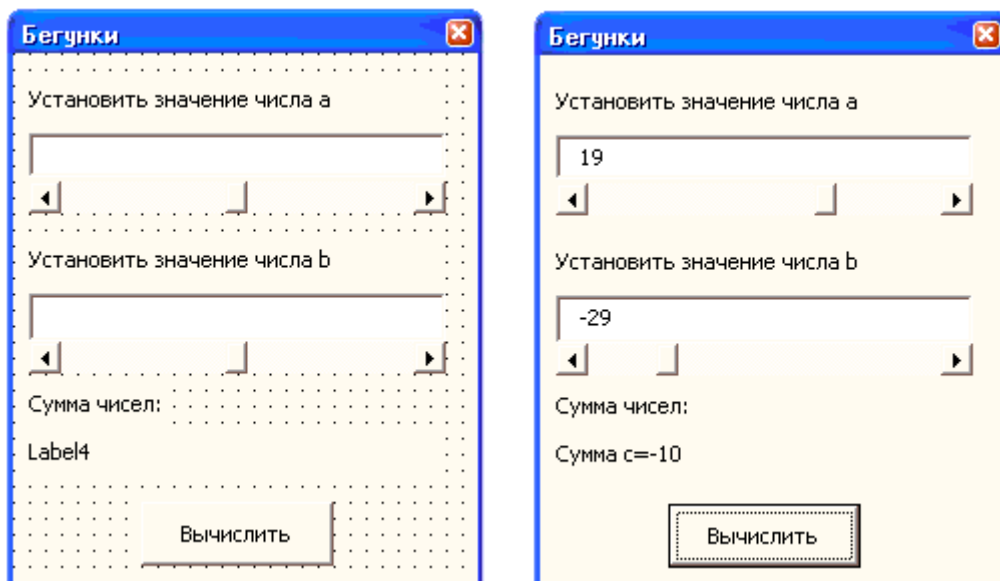


Рис. 56. Форма примера 26 в режиме конструктора и рабочем режиме

3. Обработайте элементы ScrollBar и кнопку «Вычислить».

```
Dim a, b, c As Integer
Private Sub CommandButton1_Click()
    c = a + b
    Label4.Caption = "Сумма c=" & c
End Sub
Private Sub ScrollBar1_Change()
    TextBox1.Text = ScrollBar1.Value
    a = Val(TextBox1.Text)
End Sub
Private Sub ScrollBar2_Change()
    TextBox2.Text = ScrollBar2.Value
    b = Val(TextBox2.Text)
End Sub
```

4. Откомпилируйте программу и запустите форму на выполнение.

### Счетчик SpinButton

SpinButton (счетчик, ворот) позволяет пользователю увеличивать и уменьшать числовой параметр до получения требуемого значения. Один щелчок кнопки прокрутки увеличивает или уменьшает значение свойства Value на величину, заданную свойством SmallChange. Как и для ScrollBar, интервал изменения числовой характеристики определяется значениями свойств Min и Max, вертикальная или горизонтальная ориентация счетчика – свойством Orientation, а задержка между повторными событиями Change – свойством Delay.

Чтобы изменения Value были видны пользователю, счетчик надо связать с полем ввода или с меткой в процедуре обработки события Change, так же как для полосы прокрутки.

*Пример 27.* Создать форму, содержащую два элемента spinbutton, два текстовых поля и кнопку, выводящую результат вычисления в метку на форме. Программно отражать значения в текстовых полях формы при движении бегунка. После нажатия на кнопке «Вычислить» программа считывает значения с текстовых полей, переводит эти значения в числовые и перемножает. Результат вычисления отражается в соответствующей метке на форме.

Технология выполнения

1. Активизируйте приложение Word и сохраните документ.
2. Перейдите в редактор VBA и создайте форму (рис. 57).

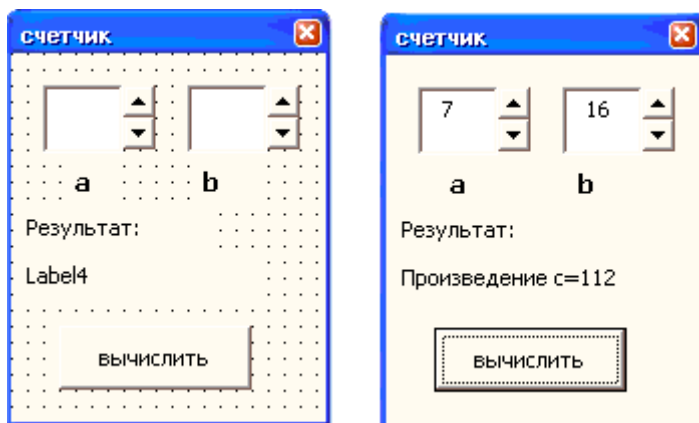


Рис. 57. Форма примера 27 в режиме конструктора и в рабочем режиме

3. Обработайте элементы SpinButton и кнопку «Вычислить».

```
Dim a, b, c As Integer
Private Sub CommandButton1_Click()
    c = a * b
    Label4.Caption = "Произведение c =" & c
End Sub
Private Sub SpinButton1_Change()
    TextBox1.Text = SpinButton1.Value
    a = Val(TextBox1.Text)
End Sub
Private Sub SpinButton2_Change()
    TextBox2.Text = SpinButton2.Value
    b = Val(TextBox2.Text)
End Sub
```

4. Откомпилируйте программу и запустите форму на выполнение.

#### 4.4. Объект DataObject[3]

##### Описание объекта DataObject

Этот объект не является элементом управления, но участвует в операциях перетаскивания выделенных текстов из одного элемента управления в другой. В нем одновременно могут храниться несколько текстовых данных в разных форматах. Когда в DataObject помещается новый текст с имеющимся в нем форматом, то прежний текст с этим форматом заменяется на новый.

Поведение DataObject похоже на поведение буфера обмена. Однако DataObject существует только в момент работы приложения и исчезает после ее завершения, а данные в буфере обмена при этом не теряются, DataObject может хранить только текстовые данные, а буфер обмена – и графические. С другой стороны, DataObject – настоящий OLE-объект и поддерживает, в отличие от буфера обмена, операции перетаскивания текста.

Текст заносится в DataObject методом SetText, а извлекается оттуда методом GetText:

объект. SetText(StoreData [, format])

и

Строка = объект. GetText([format]),

где объект – объект – владелец метода;

StoreData – текст, который надо запомнить в объекте;

format – это необязательный параметр, задающий «формат» данных (1 соответствует стандартному текстовому формату, а другие числа и строки соответствуют пользовательским форматам).

Если параметр format в вызове SetText явно не указан, то запоминаемому тексту присваивается формат стандартного текста 1. Так как для каждого формата DataObject

содержит лишь один текст с этим форматом, то фактически формат играет роль ключа, с помощью которого текст заносится и извлекается из DataObject. Метод GetFormat позволяет узнать, имеется ли в объекте DataObject текст определенного формата:

BooleanVar = объект. GetFormat(format)

Переменная BooleanVar получит значение True, если данные с указанным форматом входят в объект. Объект DataObject может обмениваться данными с буфером обмена посредством методов GetFromClipboard и PutInClipboard. Оператор

String = объект. GetFromClipboard()

помещает содержимое буфера обмена в DataObject, а вызов

объект. PutInClipboard

переносит данные из DataObject, имеющие текстовый формат 1, в буфер обмена.

*Пример 28.* Форма содержит два текстовых поля, в одном из них программно отображается текст, который при помощи методов объекта dataobject необходимо перенести (перекопировать) во второе текстовое поле. Данное действие происходит после нажатия на кнопку «Копировать», причем все производимые действия программы отображаются в соответствующей метке на форме (рис. 58).

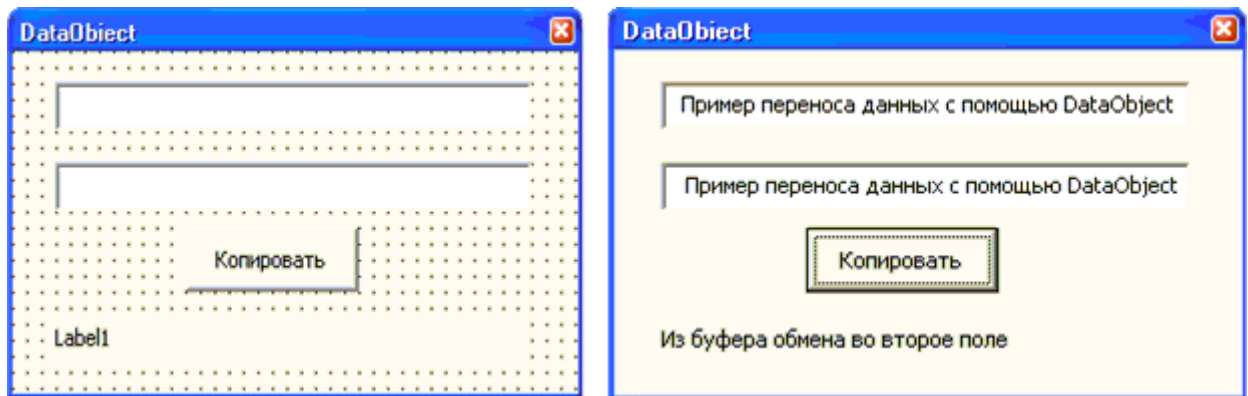


Рис. 58. Форма примера 28 в режиме конструктора и в рабочем режиме

Технология выполнения

1. Активизируйте приложение Word и сохраните документ.
2. Перейдите в редактор VBA и создайте форму (рис. 58).
3. Пропишите при инициализации окна в поле TextBox1 текстовую информацию и проинициализируйте глобальные переменные:

```
Public NewData As DataObject
```

```
Public NumClick As Integer
```

```
Private Sub UserForm_Initialize()
```

```
Set NewData = New DataObject 'инициализация объекта
```

```
NumClick = 0
```

```
'число щелчков
```

```
TextBox1.Text = "Пример переноса данных с помощью  
DataObject"
```

```
End Sub
```

4. При последовательных щелчках по командной кнопке будут происходить последовательно действия, описанные в программе:

```

Private Sub CommandButton1_Click()
Select Case NumClick
Case 0
NewData.SetText TextBox1.Text
Label1.Caption = "Из первого поля в DataObject"
Case 1
NewData.PutInClipboard
Label1.Caption = "Из DataObject в буфер обмена"
Case 2
TextBox2.Paste
Label1.Caption = "Из буфера обмена во второе поле"
End Select
NumClick = NumClick + 1
If NumClick = 3 Then NumClick = 0
End Sub

```

5. Откомпилируйте программу и запустите форму на выполнение.

*Примечание.* Перенос информации из одного поля в другое можно осуществить и через буфер обмена, минуя DataObject. Для копирования данных из поля ввода в буфер обмена можно использовать метод Copy.

### **Перемещение объектов. Реализация технологии DragAndDrop**

Для копирования данных из поля ввода в буфер обмена обычно вызывают метод Copy. Однако при организации интерфейса в диалоговых формах полезно предоставить пользователю возможность работы с техникой DragAndDrop (Переместить и Опустить). Некоторый объект захватывается мышью, перетаскивается к другому целевому объекту и отпускается, изменяя при этом свойства целевого объекта. Типичным примером является возможность перетаскивать элемент из одного списка в другой. Другой пример – перетаскивание писем и опускание их в почтовый ящик. Важным элементом этой техники является изменение внешнего вида курсора. Захват объекта происходит при подведении курсора к объекту и нажатии левой кнопки мыши. В этот момент курсор меняет внешнюю форму. Когда происходит перемещение мыши, то в тех областях, где расположен целевой объект, курсор снова меняет форму, показывая, что цель достигнута. Если в этот момент отпустить левую кнопку мыши, то операция перемещения заканчивается успешно. Если отпустить кнопку мыши в других областях, то это приведет к неудаче. Объект DataObject и его метод StartDrag являются частью этой технологии.

*Пример 29.* Создать форму, имеющую два элемента управления: список и текстовое поле. При помощи мыши из списка перетаскивают в текстовое поле необходимый элемент, выделенный в списке (рис. 59, 60, 61).

Программно заполняются элементы списка, который содержит наименования месяцев года. Программа, используя технологию DragAndDrop объекта DataObject, позволяет перетащить выбранный элемент из списка в текстовое поле выбора (рис. 60). При неправильном действии, т. е. перетаскивании не в текстовое поле, происходит вызов предупреждающего сообщения (рис. 61).

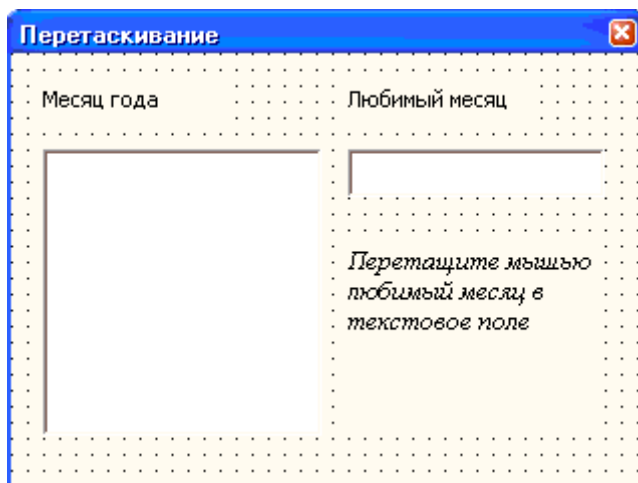


Рис. 59. Форма примера 29 в режиме конструктора

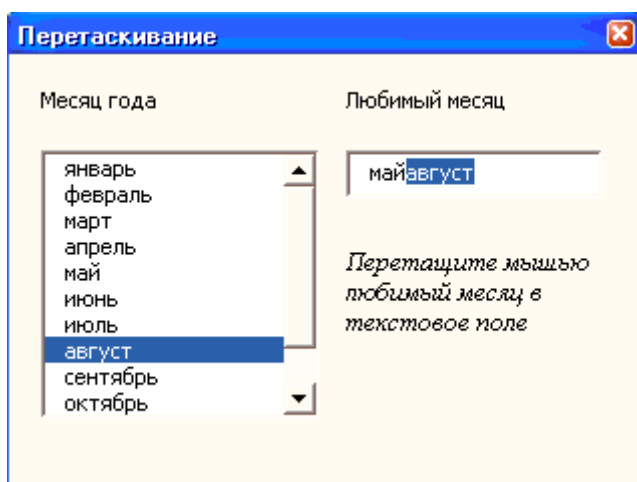


Рис. 60. Форма примера 29 в рабочем режиме

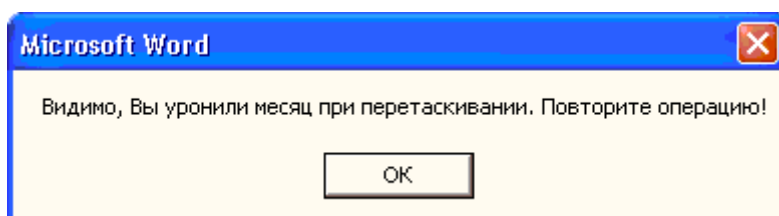


Рис. 61. Окно предупреждения о неправильно выполненной операции

Когда пользователь захватывает выбранный им элемент списка, то возникает некоторое событие. В нашем примере это событие `MouseMove`, обработчик которого и будет вызывать метод `StartDrag` объекта `DataObject`, который хранит значение перетаскиваемого элемента. Синтаксис метода:

```
Function StartDrag([Effect As fmDropEffect]) As fmDropEffect
```

Эта функция обычно вызывается в операторе присваивания вида:

```
ResultEffect=объект. StartDrag([effect as fmDropEffect])
```

Необязательный параметр `Effect` и результат выполнения функции принадлежат перечислению `fmDropEffect`. Константы, входящие в это перечисление, имеют следующие значения:

- `fmDropEffectNone = 0` – не копировать и не передвигать опущенный исходный элемент на место назначения;

- fmDropEffectCopy = 1 – копировать опущенный исходный элемент на место назначения;
- fmDropEffectMove = 2 – передвинуть опущенный исходный элемент на место назначения;
- fmDropEffectCopyOrMove = 3 – скопировать или передвинуть опущенный исходный элемент на место назначения.

Параметр Effect задает цель операции и имеет по умолчанию значение 1 (fmDropEffectCopy). Обычно он опускается, поскольку значение по умолчанию задает наиболее вероятную цель операции. Значение, возвращаемое методом StartDrag, определяет результат выполнения операции. Его можно использовать для анализа того, что же произошло в результате перетаскивания на самом деле. Между запуском метода StartDrag в правой части оператора присваивания и присваиванием результата левой части переменной ResultEffect в процессе перемещения объекта происходит много событий. Работают обработчики этих событий, и результат говорит о том, как закончился этот процесс.

Технология выполнения

1. Активизируйте приложение Word и сохраните документ.
2. Перейдите в редактор VBA и создайте форму (рис. 59).
3. Создайте обработчик события Initialize для диалогового окна, обеспечивающего

инициализацию начального состояния:

```
Private Sub UserForm_Initialize()  
With Me.ListBox1  
AddItem "январь"  
AddItem "февраль"  
AddItem "март"  
AddItem "апрель"  
AddItem "май"  
AddItem "июнь"  
AddItem "июль"  
AddItem "август"  
AddItem "сентябрь"  
AddItem "октябрь"  
AddItem "ноябрь"  
AddItem "декабрь"  
End With  
End Sub
```

4. В результате инициализируется список «Месяц года», имеющий имя ListBox1. Готовясь перетащить этот элемент в другое место, пользователь выбирает элемент этого списка. Затем он нажимает левую клавишу мыши, и у списка возникает событие MouseMove, обработчик которого имеет много параметров. Приведем текст этого обработчика:

```
Private Sub ListBox1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)  
Dim MyDataObject As DataObject  
Dim Msg As String  
Msg = "Видимо, Вы уронили месяц при перетаскивании. Повторите операцию!"  
If Button = 1 Then  
Debug.Print "MouseMove"  
Set MyDataObject = New DataObject  
Dim Effect As Integer  
MyDataObject.SetText ListBox1.Value  
Effect = MyDataObject.StartDrag(fmDropEffectCopy)  
If Effect = 0 Then MsgBox (Msg)  
Debug.Print "Effect = ", Effect  
End If  
End Sub
```

5. Откомпилируйте программу и запустите форму на выполнение.

#### **Дополнительные элементы управления**

В VBA, кроме перечисленных стандартных элементов управления, имеется ряд дополнительных. Дополнительные элементы управления являются самостоятельными объектами, обладающими как общими для всех элементов управления свойствами и



методами, так и присущими только им свойствами и методами. Для добавления дополнительных элементов управления на панель элементов необходимо:

1) выбрать команду *Сервис + Дополнительные элементы* (Tools + Additional Controls);

2) в появившемся на экране окне *Дополнительные элементы* (Additional Controls) (рис. 62) в списке *Доступные элементы* (Available Controls) установить флажок напротив добавляемого элемента;

3) нажать кнопку ОК.

Удаление ненужного элемента управления из панели элементов происходит аналогично добавлению, только флажок снимают.

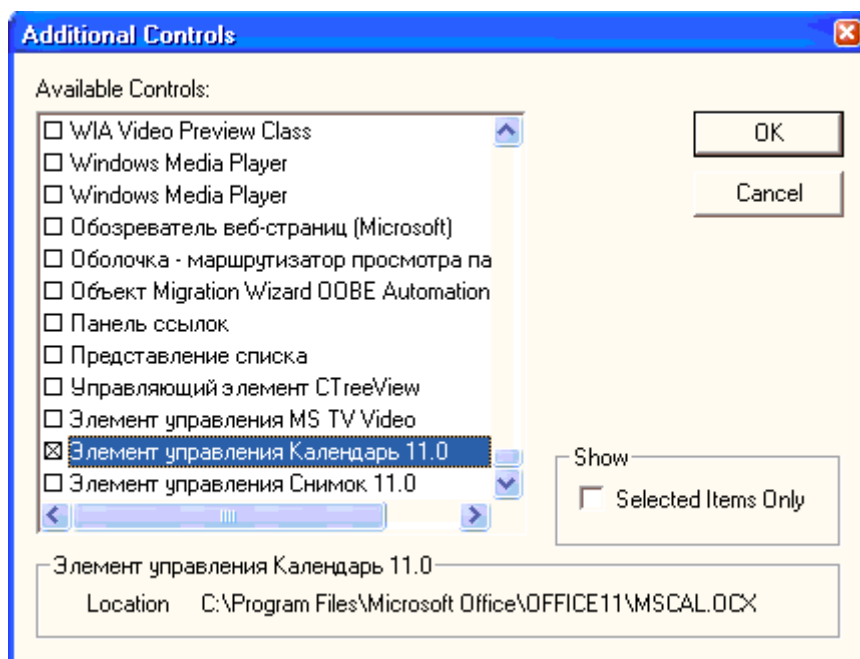


Рис. 62. Окно добавления дополнительных элементов

Среди дополнительных элементов управления очень полезным является элемент управления Calendar (календарь). Этот объект представляет средство для организации удобного интерфейса по вводу дат. Элемент управления конструируется в форме с помощью кнопки *Календарь* (Calendar).

Перечислим основные свойства элемента управления Calendar (табл. 14).

Таблица 14 -Свойства Calendar

Свойство	Действие
Day	Возвращает выбранный день
DayFont, DayFontColor	Устанавливают шрифт и цвет шрифта для названий дней недели
DayLenght	Допустимые значения: <ul style="list-style-type: none"> <li>• длинный (Long) (отображаются полные названия дней);</li> <li>• средний (Medium) (в русской версии — отображаются две буквы из названий дней, в английской — три буквы);</li> <li>• короткий (Short) (отображается только по первой букве из названия дня)</li> </ul>



Свойство	Действие
FirstDay	Первый день недели. Допустимые значения от воскресенья (Sunday) до субботы (Saturday)
Month	Возвращает выбранный месяц
MonthLenght	Допустимые значения: длинный (Long) (отображаются полные названия месяца) и короткий (Short) (отображаются только первые три буквы из названия месяца)
ShowDays	Допустимые значения: True (отображаются названия дней недели) и False (в противном случае)
ShowData-selected	Допустимые значения: True (отображается выбранная дата в верхней части календаря) и False (в противном случае)
Value	Возвращает выбранную дату
Year	Возвращает выбранный год

Перечислим основные методы элемента управления Calendar (табл. 15).

Таблица 15 -Методы Calendar

Метод	Действие
NextDay, NextWeek, NextMonth и NextYear	Устанавливает следующий день, неделю, месяц и год
PreviousDay, PreviousWeek, PreviousMonth и PreviousYear	Устанавливает предыдущий день, неделю, месяц и год
Today	Устанавливает текущую дату в календаре

В качестве примера использования календаря приведем следующую процедуру, которая считывает из календаря выбранную дату и вводит ее в ячейку рабочего листа:

```
Private Sub Calendar1_Click()
Cells(1, 1).Value = Calendar1.Value
End Sub
```

## Лабораторная работа №5. Вывод текста в документ Word

### 5.1. Основные объекты Word

#### Объект Word.Application

Дополнительно ко всем возможностям программирования на VBA, рассмотренным в части I, можно добавить возможности вывода всех результатов вычислений, преобразований, сообщений в документ Word, с возможностью дальнейшего отправления на печать. Для этого необходимо рассмотреть основные объекты приложения Word.

Ключевым в объектной модели Word является объект Application, так как он содержит все остальные объекты Word. Его элементами на разных уровнях иерархии являются около 180 объектов. Сам корневой объект Application имеет более сотни элементов: свойств, методов и событий.

#### Свойства объекта Word.Application

Свойства любого объекта делятся на две группы: свойства-участники (объекты) и терминальные свойства (обычные переменные VBA).

Единую систему организации панелей меню и инструментальных кнопок обеспечивает объект CommandBars, справоч – Assistant, поиска – FileSearch.

Центральными объектами Word являются коллекции Documents и Templates, точнее составляющие их элементы, сам документ и шаблоны.

Рассмотрим объекты второго плана.

Объект *AutoCorrect* поддерживает работу по автоматической коррекции набираемых текстов. Его возможности эквивалентны команде *Автозамена* меню *Сервис*.

Объект *Browser* позволяет перемещать точку вставки, указывающую на объекты в документе.

Коллекция объектов *Dialogs* представляет совокупность диалоговых окон, встроенных в Word. Добавлять новые или удалять элементы этой коллекции программным путем нельзя. Но соответствующие окна можно открыть и показать на экране дисплея и тем самым организовать диалог пользователем по теме, заданной соответствующим окном.

Три объекта, связанные с проверкой грамматики и орфографии: *Languages*, *Dictionaries*, *SpellingSuggestions*, – позволяют установить нужный язык, выбрать словарь, в том числе пользовательские словари, а также работать со списком слов, предлагаемых для исправления при обнаружении ошибки правописания. Команды *Правописание* и *Язык* меню *Сервис* предоставляют аналогичные, функциональные возможности при работе с документом вручную.

С помощью объекта Options можно программным путем установить различные опции приложения и документа аналогично тому, как если бы вы выбрали команду *Параметры* в меню *Сервис*.

#### Работа с документами и класс Document

Когда открывается приложение, создается коллекция документов Documents, содержащая все открытые документы. В начальный момент коллекция содержит минимум один новый или ранее существовавший документ. Новый документ добавляется методом Add, а уже существующий – методом Open объекта Documents. Чтобы добраться до нужного документа, достаточно указать его индекс – имя файла, хранящего документ, или его порядковый номер в коллекции. Для той же цели можно использовать и метод Item, но обычно он опускается. Метод Save позволяет сохранить документ, а метод Close, сохраняя документ в файле, закрывает его и удаляет из коллекции.

Глобальное свойство Dialogs возвращает коллекцию диалоговых окон. Константа wdDialogFileOpen задает конкретное диалоговое окно – объект класса Dialog.

#### Классы, задающие структуризацию текста документа

Текст – это основа большинства документов. Его можно структурировать, оперируя различными единицами при решении тех или иных задач преобразования. Минимальной

единицей текста обычно является символ. Кроме этого, существуют следующие единицы: слова, предложения, абзацы, а также более крупные образования: страницы, параграфы, главы.

Классы Characters, Words, Statements, Paragraphs, Sections позволяют работать с последовательностями (коллекциями) символов, слов, предложений, абзацев и разделов. Самой крупной единицей после абзаца выступает раздел. Элементом коллекций Characters, Words и Statements является объект класса Range. Объект Range позволяет работать как с одним элементом, так и с произвольной последовательностью элементов. Документы, поддокументы, абзацы, разделы – все они имеют метод или свойство Range, возвращающее интервал, связанный с объектом. Поэтому работа с текстом так или иначе ведется через методы и свойства объекта Range.

#### *События объекта Document*

Объект Document может реагировать на три события, возникающие в результате действий пользователя.

*Таблица 16*

События объекта Document

Событие	Пользователь
Open	Добавляет в коллекцию Documents существующий документ, открывая файл, хранящий документ
New	Создает документ
Close	Закрывает документ

#### **Документ и его части**

Рассмотрим основные классы, определяющие структуру документа.

1. *Subdocuments (Subdocument)* – коллекция и сам поддокумент. Есть некоторый разумный предел размера одного документа. Если в документе больше 10–20 страниц, работать с ним становится неудобно. В этом случае в нем выделяют главный документ и поддокументы. Главный документ в этом случае имеет коллекцию поддокументов, каждый из них является, по сути, документом, с которым можно работать независимо.

Метод *AddFromRange* класса *SubDocuments* создает поддокумент, выделяя из главного документа область, заданную параметром *Range*.

2. *Tables (Table)*, *TablesOfAuthoritiesCategories (T.O.A.C)*, *TablesOfAuthorities (TableOfAuthorities)*, *TablesOfContents (TablesOfContent)*, *TablesOfFigures (TablesOfFigure)*. Класс *Table* определяет «обычные» таблицы с произвольным количеством строк и столбцов и произвольным заполнением полей. Остальные классы задают таблицы специального вида.

3. *Shapes(Shape)*, *InlineShapes(InlineShape)* – эти две коллекции с их элементами позволяют добавлять в документ рисунки, но не только их. *ActiveX*– и *OLE*–объекты также являются элементами этих коллекций. Элементы этих двух коллекций отличаются тем, как они привязаны к документу: первые могут свободно перемещаться, вторые жестко привязаны к заданной области документа.

4. *Lists(List)*, *ListParagraphs(ListParagraph)*, *listTemplates (ListTemplate)* – списки удобно вводить в документ, когда имеешь дело с перечислением. Списки можно оформлять в соответствии с шаблоном. Существуют две группы шаблонов: нумерованные списки и списки-бюллетени. Коллекция *ListTemplates* содержит шаблоны оформления списков, а класс *ListTemplate* описывает конкретный шаблон. Шаблон применяется к списку абзацев и придает ему структуру, заданную шаблоном. Коллекция *Lists* содержит те списки документа (списки абзацев), что оформлены как нумерованные списки или списки-бюллетени. Коллекция *ListParagraphs* представляет список абзацев всех списков документа.

Свойством `ListParagraphs`, которое возвращает объект соответствующего класса, обладает не только документ, но и объекты `List` и `Range`. Так что при наличии списка – объекта `List` можно выделить список абзацев. Чаще приходится выполнять обратную операцию – применять к списку абзацев один из возможных шаблонов, придав ему «настоящую» структуру списка. Тогда используют объект `ListFormat`.

5. *Comments(Comment)*, *Bookmarks(Bookmark)*, *FootNotes (FootNote)*, *EndNotes(EndNote)*, *Fields(Field)* – эти коллекции и их элементы отражают независимые, но близкие по духу понятия. Это части документа, косвенно связанные с ним. При нормальном просмотре документа они могут быть и не видны.

- Коллекция `comments` и класс `comment` задают комментарии. Комментарии, как известно, вводятся для пояснения тех или иных терминов или понятий документа. Формально они приписываются некоторой области – объекту `range`.

- Большой документ, к отдельным частям которого приходится часто обращаться, стоит снабдить закладками. Коллекция `bookmarks` задает все закладки данного документа.

- Еще один способ комментирования – сноски. Они могут быть двух видов: подстраничные (внизу страницы) и концевые (в конце документа). Первые собраны в коллекцию `footnotes`, вторые – `endnotes`.

6. *Fields (Field)* – эта коллекция позволяет работать с полями документа. Одна из особенностей полей состоит в том, что их значения обновляются автоматически в зависимости от изменившихся внешних условий или контекста.

7. *Story Ranges (Range)* – эта коллекция представляет совокупность частей документа, называемых фрагментами (*Story*). Количество различных фрагментов документа фиксировано. Нельзя добавлять элементы в эту коллекцию обычным способом, используя метод `Add`. Фрагменты появляются в коллекции, когда создается соответствующая часть документа.

Фрагменты имеют тип, задаваемый константами из перечисления `wdStoryType`. Главный фрагмент – текст документа, тип которого задается константой `wdMainTextStory`. Комментарии, ссылки, колонтитулы составляют фрагменты других типов, т. е. сам фрагмент является объектом `Range`. Так что благодаря фрагментам можно, например, работать с коллекцией комментариев как с единой областью.

8. *Variables (Variable)* – с документом можно связать коллекцию переменных типа `Variant`. Это важная для программистов коллекция, так как время жизни переменных, в нее входящих, совпадает со временем жизни документа. Тем самым появляется возможность сохранять информацию о работе той или иной процедуры между сеансами. Например, можно иметь счетчики, подсчитывающие число вызовов макроса, и в зависимости от этого по-разному определять его дальнейшую работу.

### **Объекты Range и Selection**

Объект `Document` имеет метод `Range`, возвращающий объект `Range`, и метод `Select`, создающий объект `Selection`. Метод `Range` – это функция, возвращающая как результат объект `Range`; метод `Select` – это процедура без параметров, которая создает объект `Selection` в качестве побочного эффекта. Объект `Range` имеет метод `Select`, превращающий область объекта `Range` в выделенную. Тем самым метод `Select` определяет новый объект `Selection`. Симметрично, объект `Selection` имеет свойство `Range`, возвращающее объект `Range`, соответствующий выделенной области.

Большинство ранее описанных частей документа являются и частями (свойствами) объектов `Range` и `Selection`.

Объект `Range` напоминает матрешку: в каждую область вложена область поменьше. Вот пример корректного (хоть и не самого эффективного) задания объекта `Range`:

```
ActiveDocument.Range.Sections(1).Range.Paragraphs(1).Range.Sentences(1).Words(1).Characters(1)
```

Работа с текстом

Объекты Range и Selection позволяют выполнять основные операции над текстом: «выделить», «добавить», «заменить», «удалить». У наших объектов большой набор методов, позволяющих реализовать эти операции. Все рассматриваемые здесь методы принадлежат обоим объектам, если не сделана специальная оговорка.

#### *Выделение*

Выделить некоторую часть текста по существу означает определить объект Range или Selection. Объекты задают некоторую область в тексте документа, а их свойства Start и End позволяют установить начало и конец этой области. Меняя значения свойства, можно задать нужную область выделения.

Move является основным методом перемещения точки вставки. Остальные методы этой группы – в той или иной степени его модификации. Метод Move(Unit, Count) сжимает область в точку, стягивая ее в начало или конец, а затем перемещает точку вставки. Параметр Unit определяет единицы перемещения, а Count – количество этих единиц (по умолчанию 1). Знак переменной Count задает направление стягивания и перемещения. Положительные значения этого параметра задают стягивание к концу и перемещение вперед, отрицательные – стягивание в начало и перемещение назад. Чистое стягивание без перемещения точки вставки задается как перемещение на одну единицу. Метод возвращает количество единиц, на которое фактически произошло перемещение, или 0, если оно не осуществлено. Параметр Unit принимает значения wdCharacter (по умолчанию), wdWord, wdSentence, wdParagraph, wdSection, wdStory, wdCell, wdColumn, wdRow и wdTable.

Методы перемещения на сам текст не влияют – лишь изменяют область, заданную объектами Range и Selection. Поэтому эти методы применимы только к переменным типа Range, но не к фиксированным областям. Например, запись

ActiveDocument.Paragraphs(1).Range.Move

не имеет эффекта, поскольку область первого абзаца – вещь неизменяемая. Метод Move стягивает область в точку, которая и перемещается, поэтому после его выполнения область исчезает, остается только точка вставки. Методы MoveStart и MoveEnd перемещают начальную или конечную точку области, обычно тем самым расширяя область.

#### *Удаление текста*

Метод Delete позволяет удалить текст. Вызванный без параметров, он удаляет вызывающий его объект Range или Selection. Если он применен в форме Delete(Unit, Count), удаляется часть текста в указанной области. Параметр Unit задает единицы, но при удалении возможны только два значения: wdWord и wdCharacter. Параметр Count задает количество удаляемых единиц. Если область стянута в точку, удаляются символы перед точкой вставки или после нее в зависимости от знака параметра Count.

#### *Вставка текста*

Группа методов Insert объектов Range и Selection позволяет осуществлять вставки в документ. Для вставки текста используются методы InsertBefore(Text) и InsertAfter(Text). Параметр text типа string задает текст, вставляемый до или после области, заданной объектами range или selection. После вставки текста область автоматически расширяется, включая в себя добавляемый текст.

Свойство Text позволяет заменять текст в выделенной области, поэтому нет нужды вызывать метод Insert(Text). Методы InsertBefore и InsertAfter безопасны, так как текст добавляется, не изменяя содержимого области. Совсем иное дело – методы вставки, которые далеко не безопасны. При вставке внутрь области, например при использовании метода InsertSymbol или InsertParagraph, заменяется содержимое области.

#### *Работа с буфером*

Метод Copy, не имеющий параметров, копирует объект (содержимое области) в буфер. Метод cut, действуя аналогично, должен бы заодно и удалять объект. Но сам объект не удаляется – только стягивается в точку, так что над ним возможны дальнейшие операции.

Иногда в буфер копируют не текст, а его формат. Этим занимается метод CopyFormat, копирующий формат по первому символу объекта selection. Если этот символ – метка абзаца, копируется формат абзаца. Методом CopyFormat обладает только объект selection.

Метод Paste позволяет поместить («приклеить») содержимое буфера в область, заданную объектами Range и Selection. Эта операция опасна, так как происходит замена, а не добавление текста. Поэтому обычно метод Paste применяется к объектам Range и Selection, предварительно стянутым в точку вставки. Метод PasteFormat применяет форматирование, хранящееся в буфере, к объекту Selection.

Например, создадим макрос, который вставляет список «урок, экзамен, сдал!», копирует его и помещает еще раз на лист.

```
Sub Макрос6()  
With Selection.ParagraphFormat  
Selection.TypeText Text:="Работа с текстом:"  
Selection.TypeParagraph  
With ListGalleries(wdBulletGallery).ListTemplates(1).ListLevels(1)  
End With  
End With  
ListGalleries(wdBulletGallery).ListTemplates(1).Name = ""  
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:=ListGalleries(wdBulletGallery).ListTemplates(1)  
ContinuePreviousList = False  
ApplyTo = wdListApplyToWholeList  
DefaultListBehavior = wdWord10ListBehavior  
Selection.TypeText Text:="урок"  
Selection.TypeParagraph  
Selection.TypeText Text:="экзамен"  
Selection.TypeParagraph  
Selection.TypeText Text:="сдал"  
Selection.TypeParagraph  
Selection.Range.ListFormat.RemoveNumbers  
NumberType = wdNumberParagraph  
Selection.TypeParagraph  
Selection.Font.Bold = wdToggle  
Selection.Font.Italic = wdToggle  
Selection.TypeText Text:="Работа завершена!"  
Selection.WholeStory  
Selection.Copy  
Selection.PasteAndFormat (wdPasteDefault)  
Selection.PasteAndFormat (wdPasteDefault)  
End Sub
```

## 5.2. Форматирование документа

### Работа с текстом (продолжение)

Наиболее важной особенностью работы на VBA в Word является вставка текста в документ при работе с приложениями. Для этого служат объекты Range и Selection, которые являются главными для практически любых операций, которые можно выполнять с помощью Word VBA. Некоторые из этих действий можно применять к документам в целом, но в общем случае вам необходим диапазон или выделенная область, прежде чем вносить изменения. Мы, однако, рассмотрим действия с документом при его создании.

Открытый документ Word уже содержит объекты Range, соответствующие многим его элементам. Каждый абзац, таблица, ячейка таблицы, комментариев и т. д. определяют диапазоны. Например, для того чтобы вставить некоторый текст в уже существующий документ, необходимо прописать код:

```
ActiveDocument.Paragraphs(1).Range.Text = «Охо-хо!!»
```

Причем данная строка будет расположена в конце существующего параграфа. С другой стороны, используя объект Selection, можно также вставить некоторый текст в документ, используя метод Add и присвоение свойства Text объекту Selection:

```
If Documents.Count = 0 Then Documents.Add
```

```
Selection.Text = "Изучение работы с текстом в документе Word является важной  
составной частью умения программировать в VBA, « + TextBox1.Text +», и отвечает  
запросам всех программистов!»
```

В результате выполнения данного программного кода в документе Word будет выведена строка:

Изучение работы с текстом в документе Word является важной составной частью умения программировать в VBA, Светлана, и отвечает запросам всех программистов!

Здесь имя Светлана, например, считано с текстового поля некоторой формы, имеющей всего одно поле ввода для имени и кнопку «Вывод текста» (рис. 64).

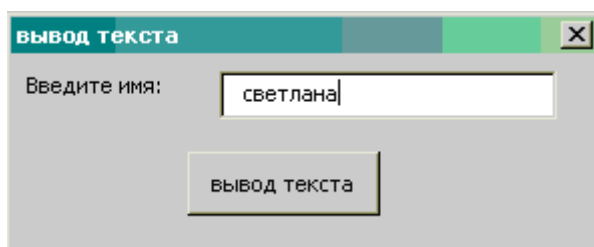
A screenshot of a Windows application window titled "Вывод текста". The window has a standard Windows XP-style title bar with a close button. Inside the window, there is a label "Введите имя:" followed by a text input field containing the text "светлана". Below the input field is a button labeled "Вывод текста".

Рис. 64. Форма ввода данных

При работе с текстом на рабочем листе Word необходимо знать следующие коды:

*определение цвета*

Selection.Font.Color =

wdColorRed – красный

wdColorDarkRed – бордовый

wdColorDarkTeal – бирюзовый

wdColorBlue – синий

wdColorGreen – зеленый

wdColorBlack – черный

wdColorOrange – оранжевый

*определение жирности*

Selection.Font.Bold =

wdToggle – жирность

*определение начертания*

Selection.Font.Italic=

wdToggle – курсив

*определение выравнивания*

Selection.ParagraphFormat.Alignment=

wdAlignParagraphRight – выравнивание по правому краю

wdAlignParagraphCenter – выравнивание по центру

wdAlignParagraphJustify – выравнивание по левому краю

*вставка в текст конкретного предложения*

Selection.TypeText Text:="Пример работы с текстом"

*вставка новой пустой строки*

Selection.TypeParagraph

*установка размера букв*

Selection.Font.Size = 14

*Примечание.* Для правильного оформления кода пользуйтесь возможностями создания макроса написания программы.

Одним из элементов оформления диалоговых окон является элемент управления *Image* (рисунок). Его основные свойства:

Autosize – изменяет размер рисунка на форме (автоматически или нет);

Picture – отображает графический файл, выводимый на форму;

pictureSizeMode – устанавливает масштабирование рисунка (не весь рисунок, вся поверхность объекта, целиком внутри объекта);

PictureAligment – устанавливает расположение рисунка внутри объекта (слева, справа, сверху, внизу);



pictureTiling – покрывает объект мозаикой из рисунка.

Данные свойства либо устанавливаются в окне свойств объекта, либо прописываются в листинге.

*Пример 30.* Создать программу, которая будет производить подсчет количества теплоты, выделяемой в проводнике при протекании в нем тока. Формула для расчета количества теплоты известна под именем закона Джоуля – Ленца:

$$Q = \frac{U^2 \cdot t \cdot S}{l \cdot p},$$

где  $Q$  – количество теплоты в Джоулях;

$U$  – напряжение в вольтах;

$t$  – время в секундах;

$S$  – площадь поперечного сечения проводника в квадратных миллиметрах;


$l$  – длина проводника в метрах;

$p$  – удельное сопротивление материала проводника в Ом · мм<sup>2</sup>/м.

Все исходные данные вводятся в текстовые поля формы. По итогам вычисления результат в виде объяснительной записки выводится в документ, а численный результат – в специальное окно формы.

Технология выполнения

Создайте форму по приведенному рис. 65.



При прохождении тока напряжением в 60 вольт по проводнику длиной 15 метров, сечением 25 кв. мм и удельным сопротивлением 15 Ом\*мм<sup>2</sup>/м за 150 секунд выделится 60 000 джоулей теплоты.

Рис. 65. Разработанная форма примера 30 в рабочем состоянии и вывод результирующих сведений в документ word

При создании формы установите необходимые свойства элементов. Установите свойство Locked элемента TextBox6 как True, чтобы не допустить случайного ввода пользователем в него текста.

Описание процедур

```
Private Sub CommandButton1_Click()
```

```
If Documents.Count = 0 Then Documents.Add
```

```
Selection.Text = "При прохождении тока напряжением в " + TextBox1.Text + "вольт  
по проводнику длиной " + TextBox4.Text + " метров, сечением " + TextBox3.Text + "кв. мм
```



и удельным сопротивлением " + TextBox5.Text + " Ом\*мм2/м за " + TextBox2.Text + " секунд  
выделится " + TextBox6.Text + « джоулей теплоты»

```
Selection.Collapse direction:=wdCollapseEnd
```

```
End Sub
```

```
Private Sub CommandButton2_Click()
```

```
Unload Me
```

```
End Sub
```

```
Private Sub TextBox1_Change()
```

```
scet
```

```
End Sub
```

```
Private Sub TextBox2_Change()
```

```
scet
```

```
End Sub
```

```
Private Sub TextBox3_Change()
```

```
scet
```

```
End Sub
```

```
Private Sub TextBox4_Change()
```

```
scet
```

```
End Sub
```

```
Private Sub TextBox5_Change()
```

```
scet
```

```
End Sub
```

```
Private Sub scet()
```

```
If IsNumeric(TextBox1.Text) = True And
```

```
IsNumeric(TextBox2.Text) = True And
```

```
IsNumeric(TextBox3.Text) = True And
```

```
IsNumeric(TextBox4.Text) = True And
```

```
IsNumeric(TextBox5.Text) = True And Not Val(TextBox4.Text)
```

```
= 0 And Not Val(TextBox5.Text) = 0 Then
```

```
rez = ((Val(TextBox1.Text) ^ 2) * Val(TextBox2.Text) *
```

```
Val(TextBox3.Text)) / (Val(TextBox4.Text) *
```

```
Val(TextBox5.Text))
```

```
TextBox6.Text = Str$(rez)
```

```
CommandButton1.Enabled = True
```

```
Else
```

```
TextBox6.Text = ""
```

```
CommandButton1.Enabled = False
```

```
End If
```

```
End Sub
```

*Примечание.* При работе данного приложения все значения полей ввода должны быть числовыми. В VBA есть специальная функция для проверки того, является ли введенная строка записью числа – IsNumeric. Для проверки отличия от нуля значений в последних двух полях ввода используется функция Val, которая переводит строковое выражение в числовое, если это строковое выражение содержит в своем начале цифры (или все состоит из них).

Функция Str делает преобразование, обратное тому, что производит Val, – конвертирует числовое значение выражения в строковое, что позволяет этому значению в дальнейшем обрабатываться как строке.

Таким образом, функции Val и Str преобразуют типы данных обрабатываемых ими переменных, соответственно, из строкового в числовой и из числового в строковый (см. главу 3).

### **Создание кнопки или панели в Word**

Назначить форме кнопку или пункт меню для непосредственного вызова приложения из Word нельзя – это можно сделать только для модулей. Поэтому нужно применить следующий способ. Создайте макрос (Меню + Insert + Module) в открытом проекте:

```
Sub counter()  
userForm1.Show  
End Sub
```

Переименуйте этот макрос по своему усмотрению, например Терло. И выполните следующие действия.

1. Щелкните правой кнопкой мыши в любом месте панели инструментов.
2. В этом меню выберите команду *Файл+параметры Word+настройки*.
3. В диалоговом окне *Настройка* щелкните на ярлыке вкладки *Команды*.
4. Прокрутите список *Категории* вниз и найдите пункт *Макросы*. Выберите этот пункт. В списке «Сохранить в» выберите имя вашего документа, в котором создан макрос.
5. В правом окне найдите необходимый макрос (либо VBA-программу).
6. Перетащите имя этого макроса или программы на панель инструментов.
7. Создайте соответствующую надпись кнопки и, если нужно, рисунок (при помощи правой кнопки мыши).
8. Закройте диалоговое окно *Настройка*, чтобы завершить работу.

По выполнении всех перечисленных пунктов будет создана кнопка запуска приложения.

#### Задачи на закрепление материала

*Пример 31.* Создать программу, позволяющую заполнять стандартные заявления студентов учебного заведения, если все необходимые данные заполняются в соответствующих текстовых полях формы, а причина пропусков занятий выбирается из раскрывающегося списка (рис. 67, 68).

Группа	АС-42	Фамилия	Петренко
Ф.И.О. кл.рук.	Сидорова О.И.	Имя	Олег
Дата пропуска	15.05.2007	Отчество	Иванович
Дата заполнения	18.05.2007	Причина	семейным обстоятельствам
заполнение			

Рис. 67. Разработанная форма примера 31 в рабочем состоянии

*Пример 32.* Создать программу, позволяющую рассчитать сумму денег, затрачиваемых на бензин при поездке на дачу.

При этом учесть следующее: условия поездки могут быть: в одну сторону, туда и обратно, можно сесть на чужую машину (данные условия учитываются в раскрывающемся списке «условия поездки» на создаваемой форме).

Полученный результат вычисляется по формуле:

Результат = расстояние \* (потребление бензина на 100 км) / 100\*цену бензина за литр.

Учесть, что если выбрали условие поездки «на чужой машине», то цена бензина равняется 0 руб.

Поездка на дачу

Расстояние, км. 55

Цена бензина (руб/литр) 18,50

Потребление бензина (литр на 100 км) 10

Условия поездки

туда и обратно

вычислить

заккрыть

полученный результат (рубли) 101,75

Рис. 69. Разработанная форма примера 32 в рабочем состоянии

*Пример 33.* Разработать программу, которая выдает *Отчет о доходе вкладчика* на настоящий момент. При этом учесть, что в интерфейсе формы процентная ставка выбирается из раскрывающегося списка (предусмотреть два вида процентов).

Доход по вкладу

Данные на вкладчика

Фамилия Евсеева

имя, отчество Ольга Константиновна

внесенная сумма 1000

Срок (дней) 150

Процентная ставка 2

Схема начисления процентов

☒ простые

☐ сложные

Вычислить

Заккрыть

На сегодняшнее число на счету находится:

1008,33

Рис. 71. Разработанная форма примера 33 в рабочем состоянии

При вычислении придерживаться следующих формул:

если схема вычисления простая, то

Доход = внесенная сумма + внесенная сумма \* процентную ставку/100\*срок/360;

если схема вычисления сложная, то

Доход = внесенная сумма + внесенная сумма \* процентную ставку/100\*срок/360 + 0,01\*(внесенная сумма \* процентную ставку/100\*срок/360).

При разработке интерфейса формы опираться на приведенный рис. 71, а при выводе расчетов в документ Word – рис. 72.

Отчет о доходе

На настоящий момент гражданин(ка) Евсеева Ольга Константиновна имеет на своем счету по простым процентам при начальной сумме вложения 1000 руб. за 150 дней при 2 %-ной ставке 1008,33 руб.

Результат вычислений с соответствующими пояснениями выводится в документ Word, а в поле формы выдается числовой результат произведенных расчетов.

*Пример 34.* Разработать программу, позволяющую заполнять заявление на восстановление студенческого билета при его порче (украли, потерял, постирал и т. д.). Причину порчи предусмотреть в раскрывающемся списке создаваемой формы.

При работе придерживаться приведенных рис. 73 и 74.

Курсивом выделены выражения, вставляемые программой после того, как будут заполнены соответствующие текстовые поля разрабатываемой формы.

Рис. 73. Разработанная форма примера 34 в рабочем состоянии

Директору БКТ  
Калгатиной Н.В.  
от студента  
Капустина Дмитрия

Заявление

Прошу заменить мне студенческий билет по той причине, что его украла.  
Сумма в количестве 100 рублей внесена в кассу (чек прилагается).

Капустин Дмитрий  
15.05.2007 г.

Рис. 74. Примерный формат вывода заявления в документ word

## Лабораторная работа №6. Автоматизация стандартных документов

### 6.1. Встроенные диалоговые окна

#### Диалоговые окна

В VBA существуют две возможности создания диалоговых окон, позволяющих вести интерактивный диалог с пользователями.

Окно сообщений MsgBox выводит простейшие сообщения для пользователя, а окно ввода InputBox обеспечивает ввод информации.

Функция InputBox выводит на экран диалоговое окно, содержащее сообщение и поле ввода, устанавливает режим ожидания ввода текста пользователем или нажатия кнопки, а затем возвращает значение типа String, содержащее текст, введенный в поле.

Синтаксис:

InputBox(сообщение [,заголовок] [,default] [,xpos] [,ypos])

Аргументы:

*сообщение* – строковое выражение, отображаемое как сообщение в диалоговом окне. Может содержать несколько строк. Для разделения строк допускается использование символа возврата каретки (chr(13)), символа перевода строки (chr(10)) или комбинации этих клавиш (chr(13) & chr(10));

*заголовок* – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку помещается имя приложения;

*Default* – строковое выражение, отображаемое в поле ввода как используемое по умолчанию, если пользователь не введет другую строку. Если этот аргумент опущен, поле ввода отображается пустым;

*Xpos* – числовое выражение, задающее расстояние по горизонтали между левой границей диалогового окна и левым краем экрана;

*Ypos* – числовое выражение, задающее расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана.

Чтобы передать эту информацию (введенное значение в поле ввода) программе, присвойте возвращенное функцией InputBox значение строковой переменной (рис. 75), например:

```
strA=InputBox(«Какие места предпочитаете?», "РЖД", "У окна")
```

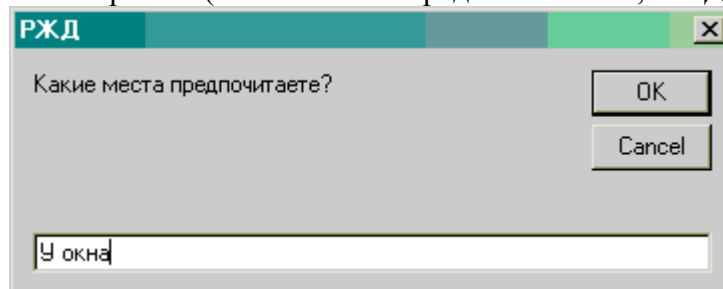


Рис. 75. Результат функции InputBox

Процедура MsgBox выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия кнопки пользователем, а затем возвращает значение типа Integer, указывающее, какая кнопка была нажата.

Синтаксис:

MsgBox(сообщение [, кнопки] [,заголовок] [,файл\_справки, раздел])

Аргументы:

*сообщение* – строковое выражение, отображаемое как сообщение в диалоговом окне;


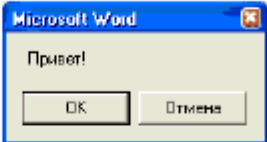
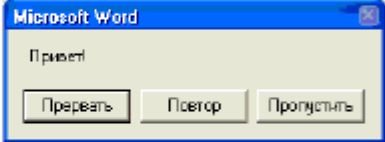
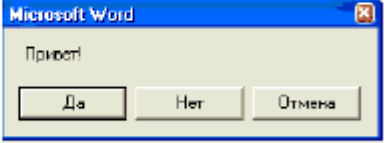
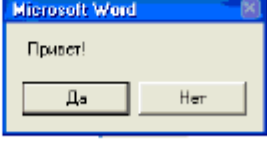
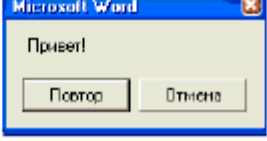
*кнопки* – числовое выражение, представляющее сумму значений, которые указывают число и тип отображаемых кнопок, тип используемого значка, основную кнопку и модальность окна сообщения. Значение по умолчанию равно 0. Все значения данного аргумента см. в табл. 17, 18;

*заголовок* – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку помещается имя приложения;

*файл справки* – строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне. Если этот аргумент указан, необходимо наличие также аргумента context;




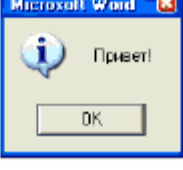
*раздел* – числовое выражение, определяющее номер соответствующего раздела справочной системы.

Таблица 17

Значения	аргумента	кнопки	процедуры	MsgBox
Константа	Значение / код	Отображаемые кнопки		
VbOkOnly	0 <code>MsgBox ("Привет!", vbOkOnly) = vbOk</code>			
VbOkCancel	1 <code>MsgBox ("Привет!", vbOkCancel) = vbOk</code>			
VbAbortRetryIgnore	2 <code>MsgBox ("Привет!", vbAbortRetryIgnore) = vbAbort</code>			
VbYesNoCancel	3 <code>MsgBox ("Привет!", vbYesNoCancel) = vbYes</code>			
VbYesNo	4 <code>MsgBox ("Привет!", vbYesNo) = vbYes</code>			
VbRetryCancel	5 <code>MsgBox ("Привет!", vbRetryCancel) = vbretry</code>			

*Примечание.* Первые кнопки активны по умолчанию.

Таблица 18

Значения	аргумента	кнопки	процедуры	MsgBox
Константа	Значение	Значок сообщения		
VbCritical	16 <code>MsgBox ("Привет!", vbCritical) = vbYes</code>		Критическое сообщение	
VbQuestion	32 <code>MsgBox ("Привет!", VbQuestion) = vbYes</code>		Предупреждающий запрос	
VbExclamation	48 <code>MsgBox ("Привет!", vbExclamation) = vbYes</code>		Предупреждающее сообщение	
VbInformation	64 <code>MsgBox ("Привет!", vbInformation) = vbYes</code>		Информирующее сообщение	

Если в окне сообщения всего две кнопки, для выяснения, на какой из кнопок был щелчок, прекрасно подходит оператор If ... then. Например:

If MsgBox («Начинать?», vbYesNo)= vbYes then

Операторы на действие этой кнопки

Else

Операторы на действие другой кнопки

End if

*Пример 35.* Создать программу таким образом, чтобы при запуске формы, вводе имени в текстовое поле и нажатии на кнопку «Вывод текста» появлялось диалоговое окно, запрашивающее разрешение вывести текст (рис. 76).



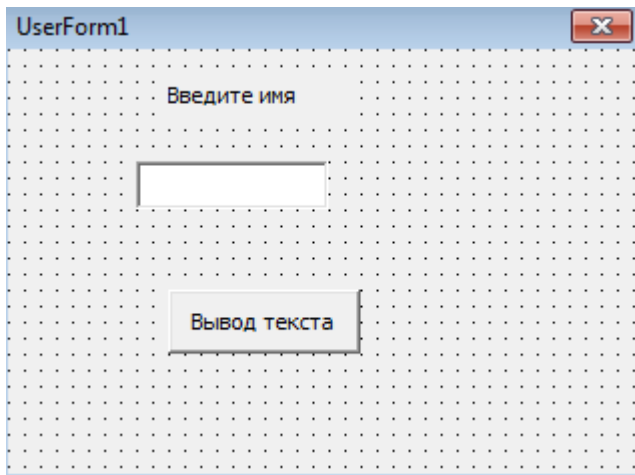


Рис. 76. Форма примера 35 в режиме конструктора

Листинг работы формы

```
Private Sub CommandButton1_Click()
    If MsgBox("Вывести текст?", vbYesNo) = vbYes Then
        Label1.Caption = "Изучение работы с текстом в документе Word является важной  
составной частью умения программировать в VBA, " + TextBox1.Text + ", и отвечает  
запросам всех программистов! "
    Else
        Unload Me
    End If
End Sub
```

В результате запуска приложения (рис. 77) в документе Word появится предложение:  
Изучение работы с текстом в документе Word является важной составной частью  
умения программировать в VBA, Света, и отвечает запросам всех программистов!

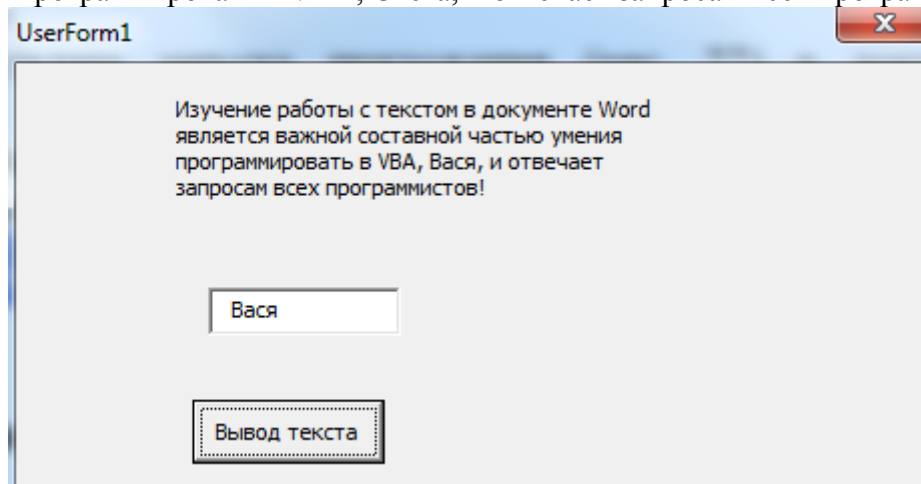


Рис. 77. Форма примера 35 в рабочем режиме

Задачи на закрепление материала

**Пример 36.** Создать форму, имитирующую простейшую игру в орла и решку (рис. 78). Игрок выставляет флажок вместо подбрасывания монеты, а компьютер после нажатия кнопки *Бросание монеты* запускает случайное число, соответствующее орлу или решке. При совпадении выигрывает компьютер, при несовпадении – игрок. Все действия сопровождаются всплывающими диалоговыми окнами.

Технология выполнения

1. Создайте форму для реализации этой игры.
2. Описание процедур.

```

Кнопка Бросание монеты
Private Sub CommandButton1_Click()
Randomize
монета = Int(2 * Rnd)
If OptionButton1.Value = True Then
If монета = 0 Then MsgBox «не везет. Займись-ка лучше изучением VBA»
If монета = 1 Then MsgBox «везунчик. Поздравляю, ты выиграл»
End If
If OptionButton2.Value = True Then

```

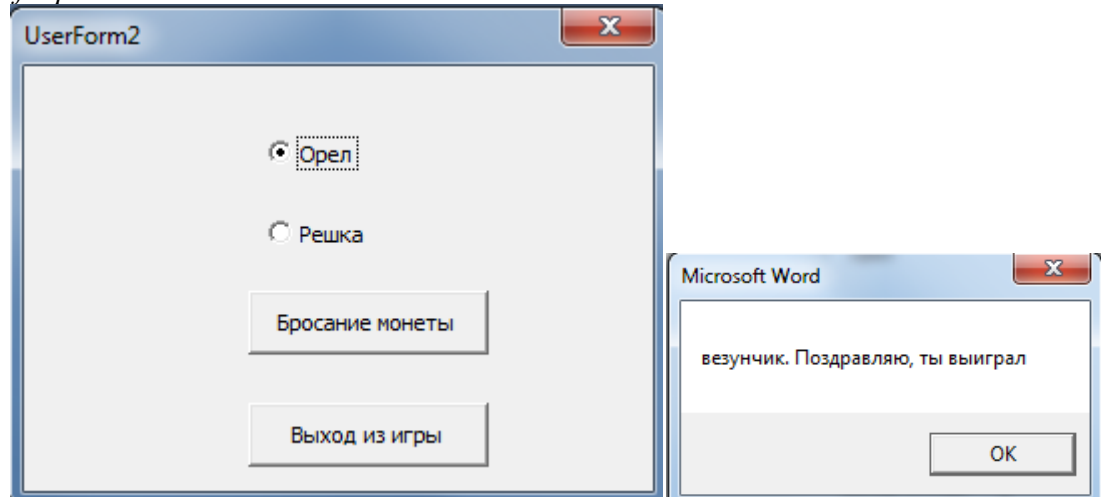


Рис. 78. Разработанная форма примера 36 в рабочем состоянии и диалоговое окно, реагирующее на результат игры

```

If монета = 1 Then MsgBox «не везет. Займись-ка лучше изучением VBA»
If монета = 0 Then MsgBox «везунчик. Поздравляю, ты выиграл»
End If End Sub
Кнопка Выход из игры
Private Sub CommandButton2_Click()
Dim ima As String
intA = MsgBox(«Нажми кнопку!», vbYesNoCancel +
vbExclamation + vbDefaultButton3, «VBA для чайников!»)
If MsgBox(«начинать?», vbYesNoCancel) = vbYes Then
ima = InputBox(«введите Ваше имя», «Пример окна ввода»)
If ima <> "" Then
MsgBox «Привет,» & ima, vbInformation, «Пример окна сообщения»
Else
MsgBox «невежа, ты забыл ввести свое имя» & ima, vbExclamation, «еще один
пример окна сообщения» End If
Else
If MsgBox(«ты точно подумал?», vbYesNoCancel) = vbNo Then
MsgBox («ха-ха»)
Else: MsgBox «Ну наконец-таки!»
Unload Me
End If
End If End Sub

```

Пример 37. В разработанном игровом приложении предусмотреть создание окна «Банк», в котором будет выводиться результат очков игрока при остановке игры. Правила игры: при выигрыше игрока добавляется единица к сумме, лежащей в банке, при проигрыше – добавляется компьютеру (отнимается из суммы, лежащей в банке).

Предусмотреть все необходимые диалоговые окна.

*Пример 38.*[4] Смоделировать полную игру в орел и решку. Игрок вносит в банк определенную сумму денег. Во время игры нельзя добавлять деньги в банк. Игра состоит из последовательности шагов, априори конечных. На очередном шаге игрок загадывает либо орел, либо решку. Компьютер «бросает» монету. Если «монета падает той же стороной», которую задал игрок, то банк увеличивается на единицу, в противном случае – уменьшается на единицу. Игра заканчивается либо по желанию игрока, либо когда величина банка становится нулем или больше 10 000 руб. (определенная сумма). Игрок забирает себе содержимое банка. Можно предусмотреть максимальные и минимальные суммы, которые были в банке в течение всей игры.

*Примечание.* Можно смоделировать бросание игральной кости, используя функцию  $\text{Int}(6 * \text{Rnd}) + 1$ . Правила меняются: выиграл тот, кто больше бросил.

Технология выполнения

Рассмотрим один из вариантов решения данной задачи. Усложним эту задачу тем, что каждое последующее окно вызывается соответствующим действием.

1. Пусть при запуске приложения появляется первое диалоговое окно (рис. 79). При нажатии на кнопку «Начать игру» появляются диалоговые окна, запрашивающие имя игрока и подтверждение начать игру (см. рис. 80). После чего появляется форма ввода ставок (см. рис. 81).

```
Private Sub CommandButton1_Click()  
    mya = InputBox(«введите ваше имя», «Регистрация», «????»)  
    If MsgBox(«Начинать?», vbYesNo, «Вы не передумали?») =  
        vbYes Then  
        UserForm2.Show  
    Else  
        UserForm4.Show  
    End If  
End Sub
```

Данная форма производит начальный выбор ставки и запускает главную форму примера 38.

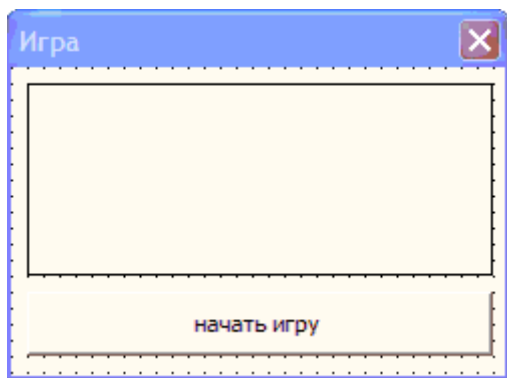


Рис. 79. Диалоговое окно запуска игры примера 38

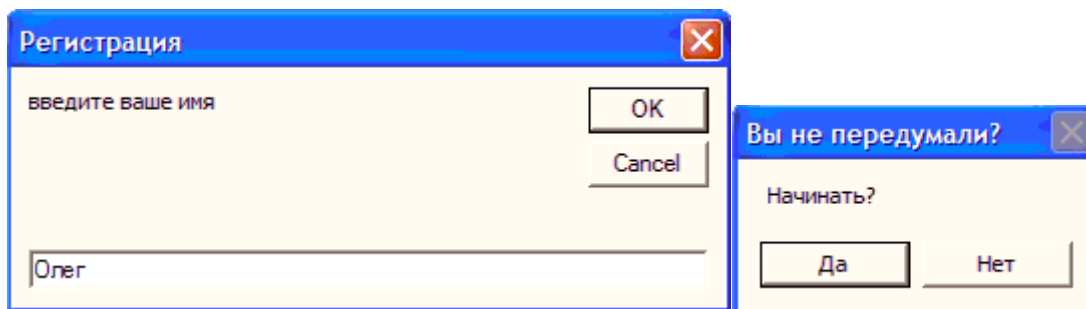


Рис. 80. Диалоговые окна начала игры

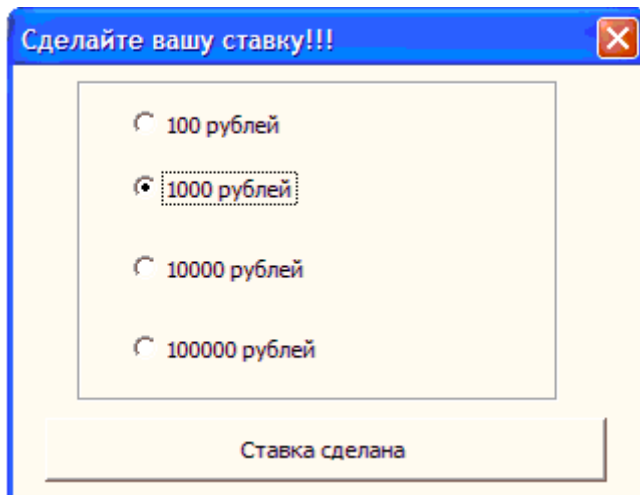


Рис. 81. Диалоговое окно выбора ставки

Кнопка *Бросок* имитирует подбрасывание монеты игроком, производит расчет выигрыша и проигрыша игрока, выводя соответствующие значения в текстовые окна с сообщением счета игры (рис. 82).

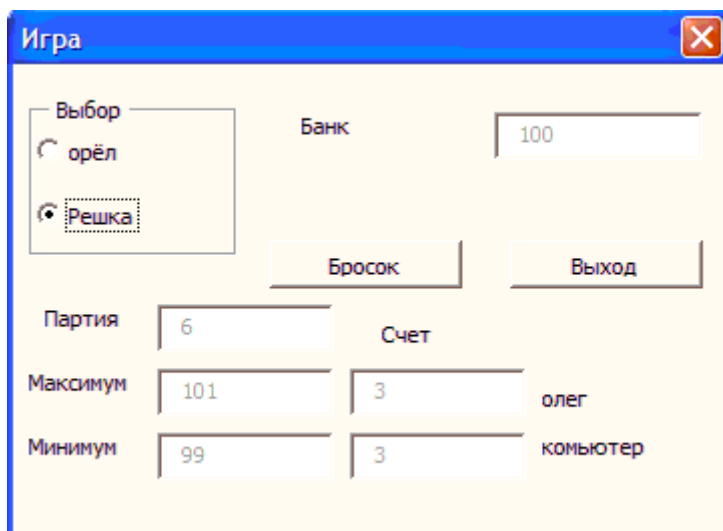


Рис. 82. Главная форма примера 38, имитирующая игру в орел и решку

*Кнопка Бросок*

```
Private Sub CommandButton1_Click()
    TextBox1.Value = TextBox1.Value + 1
    If b = Fix(Rnd * 2 + 1) Then
        TextBox4.Value = TextBox4.Value + 1
        TextBox5.Value = TextBox5.Value + 1
    Else
        TextBox4.Value = TextBox4.Value - 1
        TextBox6.Value = TextBox6.Value + 1
    End If
    If TextBox4.Value < 1 Then
        MsgBox («Вы проиграли!!!»)
        UserForm4.Show
    End If
    If Val(TextBox2.Text) < Val(TextBox4.Text) Then
```

```

    TextBox2.Value = Val(TextBox4.Text)
Else
    If Val(TextBox3.Text) > Val(TextBox4.Text) Then
        TextBox3.Value = Val(TextBox4.Text)
    End If
End If
    OptionButton1.Value = False
    OptionButton2.Value = False
    CommandButton1.Enabled = False
End Sub
Кнопка Выход завершает игру, выдает итоговые сообщения (см. рис. 83).
Private Sub CommandButton2_Click()
    MsgBox («Партий» + TextBox1.Value + (Chr(13)) + «в банке „ + TextBox4.Value +
(Chr(13)) + „ваш максимум“ + TextBox2.Value + (Chr(13)) + „ваш минимум“ +
    TextBox3.Value + (Chr(13)) + „счетом“ + TextBox5.Value + “:» + TextBox6.Value)
    UserForm4.Show
End Sub
Private Sub UserForm Initialize()
    Unload UserForm2
    OptionButton1.Value = True
    TextBox4.Value = a
    Label6.Caption = imya
    TextBox2.Value = TextBox4.Value
    TextBox3.Value = TextBox4.Value
End Sub

```

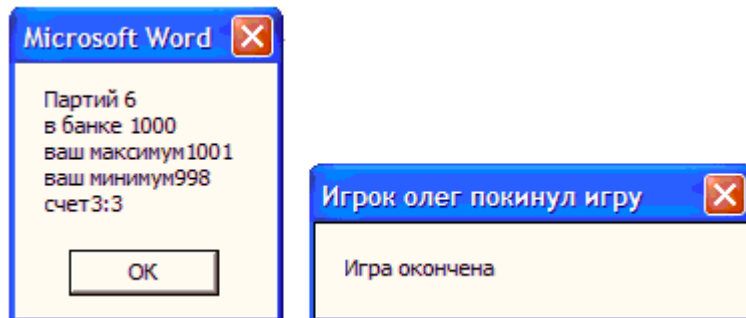


Рис. 83. Итоговые расчеты игры примера 38

*Примечание.* Для корректной работы игры необходимо создать модуль, в котором объявить глобальные переменные, содержащие сведения о значении начальной ставки и имени игрока, а также запускающий первую форму игры при помощи кнопки на панели инструментов. Кроме того, необходимо доработать все модули для соответствующих форм (рис. 84).

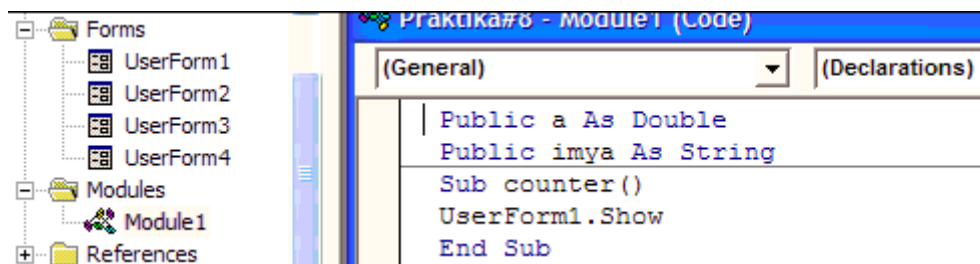


Рис. 84. Объявление глобальных переменных и создание модуля запуска главной формы игры в примере 38

## Лабораторная работа №7. Работа с элементами active x

В наших работах мы подробно рассмотрим практические примеры, связанные с использованием программирования, которое заложено в Microsoft Office. Оно основывается на языке VBA, название которого в полном виде выглядит так Visual Basic for Application. Учитывая, что все рассматриваемые разработки будут использовать VBA, наша первоочередная задача познакомиться с данным языком.

Примеры этой темы предназначены для категории читателей, которая либо с VBA не встречалась, либо это знакомство было весьма поверхностным. Все последующие главы будут полностью посвящены рассмотрению практических офисных задач, и, таким образом, эта работа является для них базовой.

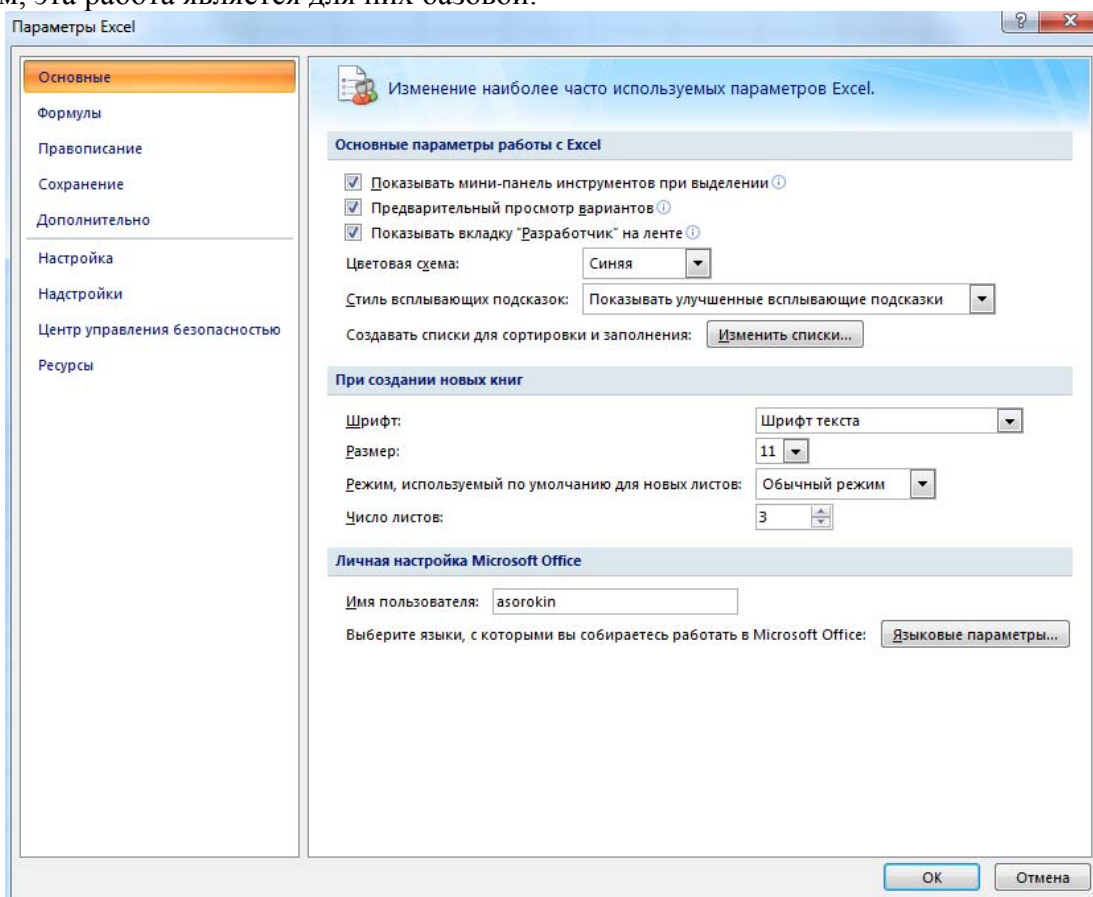


Рис. 1 Закладка Основные окна Параметры Excel

Предполагается, что читатель хотя бы в незначительной степени знаком с чисто пользовательскими возможностями Microsoft Excel прошлых версий. Важно отметить, что наибольший эффект будет наблюдаться, если запустить Microsoft Excel параллельно с чтением наших статей и выполнять все описанные разработки на компьютере. При рассмотрении примеров на протяжении всего изучения нам понадобятся элементы ActiveX, которые по-другому можно назвать Элементы управления.

И в качестве первого шага необходимо в окне щелкнуть кнопкой Параметры Excel. В результате перед вами откроется новое окно с набором закладок. Здесь на закладке Основные следует сделать установку для отображения на ленте вкладки Разработчик, что реализуется с помощью флажка Показывать вкладку «Разработчик» на ленте.

В Microsoft Excel элементы ActiveX можно размещать и на пользовательских формах, и на самих рабочих листах (на верхнем графическом слое, где располагаются рисунки, диаграммы и другие объекты). Часто вставка нескольких элементов ActiveX значительно упрощает работу с данными на рабочем листе. Все элементы управления делятся на две группы: элементы управления формами и элементы управления ActiveX. Оба набора элементов управления имеют свои преимущества и недостатки. В общем случае



элементы управления формами проще в применении, но зато элементы ActiveX являются более гибкими. В наших примерах мы будем использовать исключительно элементы ActiveX.

В результате окно приложения Microsoft Excel дополнится новыми пиктограммами, которые нам потребуются в дальнейшем (рис. 2). Среди тех, которыми мы будем активно пользоваться на протяжении всей книги, отметим пиктограмму с надписью Вставить. С ее помощью на рабочем листе можно размещать элементы ActiveX, которые существенно дополняют функциональность книг Microsoft Excel. Они типичны для различных приложений Windows (речь идет о кнопках, полосах прокрутки, текстовых окнах, переключателях, списках и т. д.). Кроме пиктограммы Вставить мы будем активно пользоваться и другими присутствующими на ленте Режим конструктора, Свойства, Visual Basic и Макросы.

Перед тем как начать какие-либо действия с тем или иным элементом ActiveX, его необходимо поместить на рабочий лист. Это легко выполнить, если воспользоваться пиктограммой Вставить. На рис. 3 показано размещение на рабочем листе элемента ActiveX Кнопка.

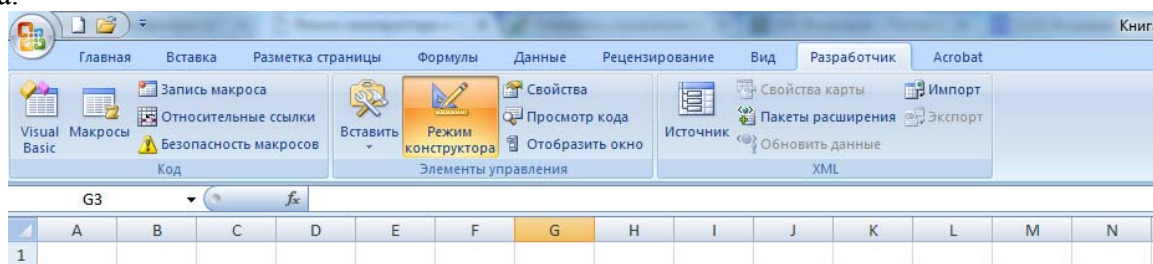


Рис. 2. Отображение вкладки Разработчик на ленте

При наведении курсора мыши на пиктограмму элемента ActiveX Кнопка в качестве подсказки появляется еще один вариант его названия: элемент управления «Кнопка». Далее по тексту мы будем использовать оба варианта названия — элементы ActiveX и элементы управления.

Для того чтобы перенести элемент ActiveX на рабочий лист, необходимо щелкнуть на панели инструментов на его пиктограмме левой кнопкой мыши и далее переместить курсор мыши на рабочий лист. Курсор мыши примет вид, аналогичный математическому знаку «плюс». Теперь следует нажать левую кнопку мыши и, не отпуская ее, переместить мышь вправо и вниз, а затем отпустить ее левую кнопку. В результате на рабочем листе мы увидим изображение появившегося элемента ActiveX, окруженного маркерами (рис. 3).

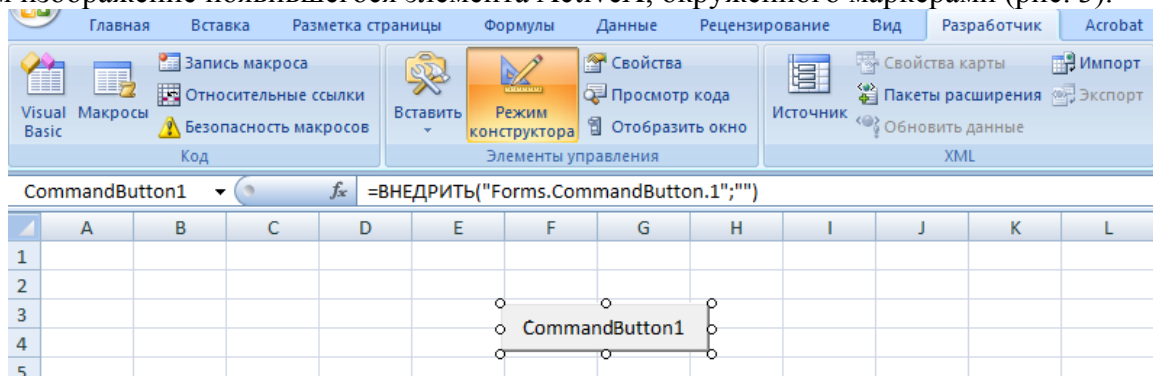


Рис. 3. Размещение кнопки на рабочем листе в режиме конструктора

После размещения элемента ActiveX на рабочем листе активизируется кнопка Режим конструктора, которая позволяет пользователю далее работать в одном из двух режимов. Один из них так и называется режим конструктора. В этом случае можно работать с элементами ActiveX для создания необходимого интерфейса на рабочем листе. Также в этом режиме пользователю предоставляется возможность создавать программные разработки.

Появление маркеров (см. рис. 3), окружающих элемент ActiveX, как раз и является признаком того, что мы работаем в режиме конструктора (кнопка Режим конструктора в этом случае выглядит нажатой). На начальном этапе создания разработки работа в режиме конструктора сводится к размещению элементов управления на рабочем листе и изменению их свойств. Далее производится программирование различных событий, связанных с элементами управления, листами и книгой в целом.

Важно отметить, что после размещения элемента ActiveX на рабочем листе, в нашей книге (можно даже сказать — на данном листе книги) появился новый элемент, который фактически представляет собой «программный» объект. Аналогичным образом на рабочих листах книг Excel можно размещать и другие элементы ActiveX.

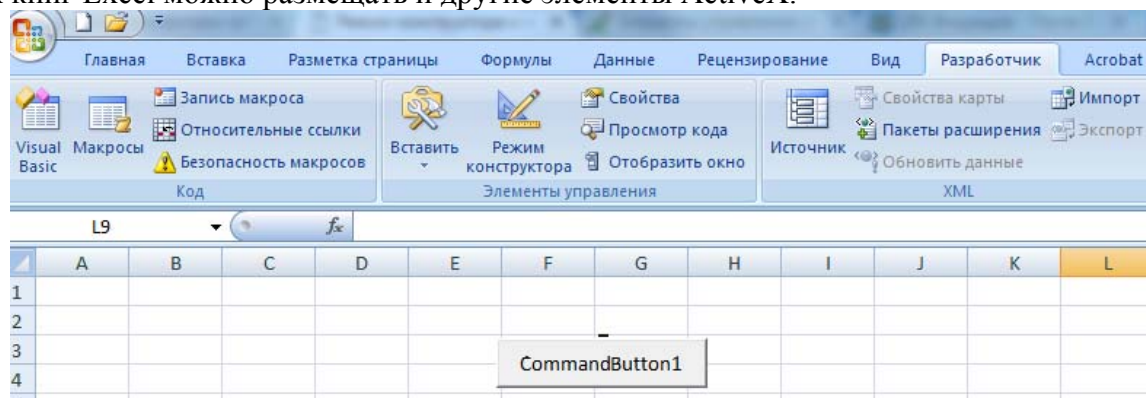


Рис. 4. Вид рабочего листа при выходе из режима конструктора

Другой режим можно назвать режимом выполнения или режимом работы (какого-то однозначно принятого названия не существует). Чтобы перейти в этот режим (то есть выйти из режима конструктора), необходимо отжать кнопку Режим конструктора (рис. 4). После этого в нашем случае можно просто пощелкать кнопкой, которую мы создали на рабочем листе (функциональные действия, которые можно обеспечить в этом случае, мы рассмотрим в дальнейшем). Далее опять вернемся в режим конструктора. С помощью маркеров, окружающих кнопку, можно легко изменить ее размеры. Также с помощью мыши можно перемещать созданную кнопку по рабочему листу.

Как уже было сказано во введении, у программных объектов имеются различные свойства. Первое очевидное желание заключается в том, чтобы просмотреть (а затем и изменить) значения свойств объектов. Для просмотра свойств объекта его необходимо сначала выделить в режиме конструктора (щелкнуть на нем мышью), а затем с помощью пиктограммы Свойства (она расположена рядом с пиктограммой Режим конструктора) открыть окно свойств (рис. 5).



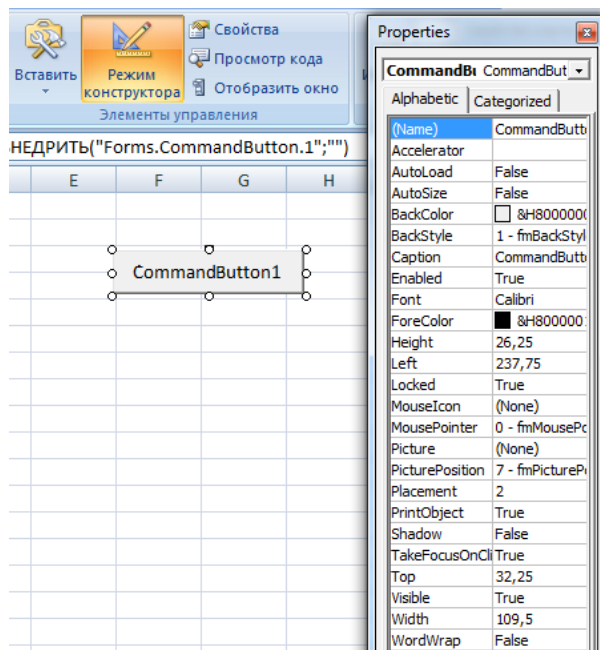


Рис. 5. Окно свойств объекта

В окне свойств отображается таблица набор строк в два столбца. При этом левый столбец отводится для названий свойств, а в правом расположены значения этих свойств. Понятно, что содержание левого столбца изменить нельзя свойства объекта уже определены его разработчиком (компанией Microsoft). А значения свойств мы поменять можем для этого достаточно щелкнуть в ячейке правого столбца и изменить ее содержание.

Если приглядеться к окну на рис. 5, то мы увидим, что для ряда свойств возможны только два варианта значений. Например, для свойства Visible (определяет видимость объекта на экране) возможны только значения False и True. Такая ситуация наблюдается и для ряда других свойств. Среди них можно отметить Enabled определяет доступ к объекту в режиме выполнения (если установить для этого свойства значение False и выйти из режима конструктора, то объект будет недоступен). Свойство AutoSize позволяет обеспечить автоматическую подгонку размеров кнопки под размер текста, расположенного на ней. Такое будет происходить, если установить True в качестве значения данного свойства.

Для некоторых других свойств следует устанавливать числовые значения. Так, местоположение и размеры элемента управления определяются следующими свойствами:

- Width ширина;
- Height высота;
- Top координата верхней границы элемента, начиная от верхней части листа;
- Left координата левой границы элемента, начиная от левой части листа.

Если для свойства Enabled установлено значение False, то элемент управления на экране будет выглядеть более блекло (такая ситуация знакома по работе с различными приложениями, когда ряд разделов меню недоступен).

Если теперь в режиме конструктора вы будете перемещать созданную на листе кнопку по экрану и изменять ее размеры (с помощью окружающих ее маркеров), то соответствующие значения в окне свойств будут меняться. И наоборот, изменив значения в окне свойств, вы увидите изменения на экране. Для установки значений ряда свойств следует использовать пиктографические меню. Так, с помощью свойства BackColor можно изменять цвет кнопки. Для этого всего лишь требуется щелчком мыши выбрать необходимый цвет. Аналогичное свойство ForeColor определяет цвет текста на кнопке. За надпись на кнопке отвечает другое свойство Caption. Если напротив названия этого свойства ввести текст, то мы увидим его и на кнопке.

Для изменения шрифта текста на кнопке имеется свойство Font. Если в окне свойств попытаться изменить его значение, то перед вами откроется знакомое (но работе с

различными приложениями) окно для выбора шрифта и размера букв. При этом технические действия здесь выполняются с помощью щелчков мышью.

Еще одно интересное свойство, `Picture`, позволяет разместить на кнопке изображение (рис. 1.12) из графического файла. Для этого в качестве значения свойства следует указать имя файла. Это действие производится с помощью стандартного диалогового окна (рис. 1.13), в котором необходимо выбрать один из графических файлов на компьютере либо в сети. В случае, если вы захотите убрать изображение, то в поле для значения данного свойства следует воспользоваться клавишей `Delete`.

Во введении уже упоминалось, что основное свойство объекта это имя. В окне свойств оно называется `Name`. Как мы видели на примере создания кнопки, Excel автоматически присвоил ей имя. Имя первой созданной кнопки на рабочем листе по умолчанию `CommandButton1`, которое вы при желании можете изменить. Явной необходимости изменения имен, которые присваиваются по умолчанию, нет. Однако очень часто программисты назначают объектам свои имена. Это связано с тем, что к любым объектам на листе мы можем получить доступ из текста программы (фактически требуется обращаться к свойствам и методам этих объектов), указав имя конкретного объекта. В связи с этим с точки зрения организации программного кода часто удобнее использовать свою систему назначения имен объектов.

Система формирования имен по умолчанию заключается в комбинации типа элемента управления (`CommandButton` — командная кнопка) и числа (порядкового номера элемента управления данного типа).

Для первой созданной кнопки имя — `CommandButton1`. Если мы на рабочем листе разместим еще одну кнопку, то она будет иметь имя `CommandButton2`. При добавлении последующих кнопок данный принцип сохраняется. Если вы все же решитесь изменить имя, то вместо `CommandButton1` в окне свойств следует набрать слово или словосочетание, которое вас устраивает. Но заметим, что оно должно быть без пробелов внутри (частая ошибка в первых разработках). В качестве примера подберем новое имя для кнопки — `PrimerButton`.

Теперь, если вы произвели описанные выше установки (а также убрали изображение очистили значение в свойстве `Picture`). Для свойства `AutoSize` здесь установлено значение `True`, что позволяет подогнать размеры кнопки под расположенную на ней надпись.

С подобных шагов начинается любая разработка на листе размещаются элементы `ActiveX`, и ряду их свойств присваиваются необходимые значения. На первый взгляд свойства `Name` и `Caption` похожи, и кажется, одно из них является лишним. Однако это не так, и в связи с этим дадим небольшое пояснение. Дело в том, что `Name` внутреннее название объекта, и значение этого свойства используется на программном уровне (во введении об этом уже шла речь). Если мы хотим использовать в программном коде обращение к свойствам и методам объекта, то в строке программной процедуры необходимо указать имя объекта. Свойство же `Caption` отвечает просто за надпись на кнопке, и значение его только отображается на экране.

Реально у каждого элемента `ActiveX` имеется много свойств. Некоторые из них являются общими для большинства (или даже для всех) элементов `ActiveX`, а другие свойства уникальны для определенных элементов управления.

Таким образом, результатом выполненных действий явились создание нового объекта в нашей рабочей книге Microsoft Excel и установка значений для ряда его свойств. Вообще, в режиме конструктора мы размещаем на рабочем листе элементы `ActiveX` и устанавливаем значения их свойств, а в режиме выполнения работаем с созданными объектами (щелкаем кнопками, вводим текст в текстовые окна и т. д.).

## Лабораторная работа №7. Особенности использования VBA в Excel

### 7.1. Основные объекты VBA в Excel

#### Использование объектов Range и Selection

В Excel наиболее важным является объект Application. Объект Application (приложение) является главным в иерархии объектов Excel и представляет само приложение Excel. Он имеет более 120 свойств и 40 методов. Эти свойства и методы предназначены для установки общих параметров приложения Excel. В иерархии Excel объект Workbook (рабочая книга) идет сразу после объекта Application и представляет файл рабочей книги. Рабочая книга хранится либо в файлах формата XLS (стандартная рабочая книга), либо XLA (полностью откомпилированное приложение). Свойства и методы рабочей книги позволяют работать с файлами. Однако наиболее «употребляемым» на практике является объект Range, который наилучшим образом отображает возможности использования VBA в Excel (о свойствах объекта Range см. табл. 19, о методах – табл. 20).

В иерархии Excel объект *Range* (диапазон) идет сразу после объекта *worksheet*. Объект *Range* является одним из ключевых объектов VBA. Объект *selection* (выбор) возникает в VBA двояко – либо как результат работы метода *Select*, либо при вызове свойства *selection*. Тип получаемого объекта зависит от типа выделенного объекта. Чаще всего объект *Selection* принадлежит классу *Range*, и при работе с ним можно использовать свойства и методы объекта *Range*. Интересной особенностью объектов *Range* и *Selection* является то, что они не являются элементами никакого семейства объектов.

При работе с объектом *Range* необходимо помнить, как в Excel ссылаются на ячейку рабочего листа.

#### Задание групп строк и столбцов с помощью объекта Range

Если в диапазоне указываются только имена столбцов или строк, то объект *Range* задает диапазон, состоящий из указанных столбцов или строк. Например, *Range* («a: c») задает диапазон, состоящий из столбцов a, в и c, а *Range*(«2:2») – из второй строки. Другим способом работы со строками и столбцами являются методы *Rows* (строки) и *columns* (столбцы), возвращающие коллекции строк и столбцов. Например, столбцом a является *columns* (1), а второй строкой – *Rows* (2).

#### Связь объекта Range и свойства Cells

Так как ячейка является частным случаем диапазона, состоящим только из единственной ячейки, объект *Range* также позволяет работать с ней. Объект *Cells* (ячейки) – это альтернативный способ работы с ячейкой. Например, ячейка A2 как объект описывается *Range* («A2») или *Cells* (1, 2). В свою очередь, объект *cells*, вкладываясь в *Range*, также позволяет записывать диапазон в альтернативном виде, который иногда удобен для работы, а именно *Range*(«A2:C3») и *Range*(*Cells*(1,2), *Cells*(3,3)) определяют один и тот же диапазон.

Таблица 19 Свойства объекта Range

Свойство	Действие
Value	Возвращает значение из ячейки или в ячейки диапазона. В данном примере переменной x присваивается значение из ячейки c1: x = Range("C1").Value В следующем примере в диапазон a1:b2 введена 1: Range("A1:B2").Value = 1
Name	Возвращает имя диапазона. В данном примере диапазону a1:b2 присваивается имя «итоги»: Range("A1:B2").Name = "Итоги"

Свойство	Действие
Count	Возвращает число объектов в наборе. В данном примере переменной x присваивается значение, равное числу строк диапазона a1:b2: x = Range("A1:B2").Rows.Count
ColumnWidth, RowHeight	Возвращает ширину столбцов и высоту строк диапазона
CurrentRegion	Возвращает число строк текущего диапазона. Текущим является диапазон, ограниченный пустыми строками и столбцами и содержащий данный элемент. В следующем примере переменной y присваивается значение, равное числу строк в текущем диапазоне, содержащем ячейку a1: y = Range("A1").CurrentRegion.Rows.Count
WrapText	Позволяет переносить текст при вводе в диапазон. Допустимые значения True и False. В следующем примере в ячейку b2 вводится текст «длинный текст», и в этой ячейке устанавливается режим ввода текста с переносом: With Range("B2").Value = "Длинный текст".WrapText = True End With
EntireColumn, EntireRow	Возвращает столбец и строку соответственно. В данном примере очищается содержимое строки и выделяется столбец с активной ячейкой: ActiveCell.EntireRow.Clear ActiveCell.EntireColumn.Select
Comment	Возвращает объект comment (примечание), который связан с левым верхним углом диапазона при отображении на экране. Объект comment является элементом семейства comments. Метод AddComment, примененный к диапазону, создает новое примечание. Среди методов объекта Comment отметим только метод Text, который задает текст, выводимый в примечании. Синтаксис: Text(Text, Start, Overwrite) Здесь Text — строка, выводимая в качестве примечания; Start — с какого символа вводится текст в уже существующее примечание. Если аргумент опущен, то из примечания удаляется весь ранее введенный текст; overwrite — допустимые значения: True (вводимый текст записывается поверх уже существующего) и False (вводимый текст вставляется в уже существующий). Среди свойств объекта comment отметим только свойство visible, устанавливающее отображение примечания при активизации диапазона, имеющего определенное примечание

Свойство	Действие
Vertical Alignment	Вертикальное выравнивание. Допустимые значения: xlBottom (выравнивание по нижнему краю); xlCenter (выравнивание по центру); xlJustify (выравнивание по высоте); xlTop (выравнивание по верхнему краю)
Orientation	Ориентация. Допускается либо угол поворота текста в градусах от $-90^{\circ}$ до $90^{\circ}$
ShrinkToFit	Допустимые значения: True (автоматическое изменение шрифта так, чтобы текст помещался в ячейку) и False (в противном случае)
ShrinkToFit	Допустимые значения: True (автоматическое изменение шрифта так, чтобы текст помещался в ячейку) и False (в противном случае)
Font	Возвращает объект Font (шрифт). Объект Font имеет следующие свойства: Name — строка, указывающая имя шрифта, например «Arial Cyr»; FontStyle — стиль, возможен Regular (обычный), Bold (полужирный), italic (курсив), Bold italic (полужирный курсив); size — размер; strikethrough — допустимы два значения: True (буквы имеют линию по центру, как будто они перечеркнуты) и False (не имеют линии по центру); superscript — допустимы два значения: True (текст используется как верхний индекс) и False (не используется как верхний индекс); Subscript — допустимы два значения: True (текст используется как нижний индекс) и False (не используется как нижний индекс); underline — устанавливает тип подчеркивания, допустимыми являются значения: xlNone (нет подчеркивания), xlSingle (одинарное по значению)
Horizontal Alignment	Горизонтальное выравнивание. Допустимые значения: xlGeneral (обычное выравнивание, зависящее от типа вводимых значений); xlCenter (выравнивание по центру); xlRight (выравнивание по правому краю); xlLeft (выравнивание по левому краю); xlJustify (выравнивание по ширине); xlCenterAcrossSelection (выравнивание по центру в выделенном диапазоне); xlFill (выравнивание по ширине)

Таблица 20

Методы	объекта	Range
Метод	Действие	
AutoFit	Автоматически настраивает ширину столбца и высоту строки	
Clear, ClearComments, ClearContents, ClearFormats	Метод clear очищает диапазон. В следующем примере очищается диапазон A1:G37. Range("A1:G37").Clear Методы ClearComments, ClearContents, ClearFormats и ClearNotes очищают в указанном диапазоне	
Insert	Вставка ячейки или диапазона ячеек. В следующем примере вставляется новая строка перед четвертой строкой рабочего листа Лист1: Worksheets("Лист1").Rows(4).Insert	
Select	Выделение диапазона	
ClearNotes	Комментарии, содержание, форматы и примечания соответственно	
Copy	Копирует диапазон в другой диапазон или в буфер обмена. Синтаксис: Copy(destination) Аргумент destination определяет диапазон, куда копируется данный диапазон. Если аргумент destination опущен, то копирование происходит в буфер обмена. В данном примере диапазон a1:D4 рабочего листа копируется в диапазон E5 листа 2: Worksheets("Лист1").Range("A1:D4").Copy destination:=Worksheets("Лист2").Range("E5")	
Delete	Удаляет диапазон	
AddComment	Добавляет примечание к диапазону. Синтаксис: AddComment (Text) Text — строковое выражение, добавляемое в качестве примечания	
Address	Возвращает адрес ячейки. Синтаксис: Address(rowAbsolute, columnAbsolute, referenceStyle, external, relativeTo) Аргументы: rowAbsolute — допустимы два значения True и False, если используется значение True или аргумент опущен, то возвращается абсолютная ссылка на строку; columnAbsolute — допустимы два значения True и False, если используется значение True или аргумент опущен, то возвращается абсолютная ссылка на столбец;	

Метод	Действие
	<p>referenceStyle — допустимы два значения xlA1 и xlR1d, если используется значение xlA1 или аргумент опущен, то возвращается ссылка в виде формата A1;</p> <p>external — допустимы два значения True и False, если используется значение False или аргумент опущен, то возвращается относительная ссылка.</p> <p>Следующий пример показывает различные результаты адресации.</p> <p>MsgBox Cells(1, 1).Address</p> <p>В диалоговом окне отображается адрес \$A\$1.</p> <p>MsgBox Cells(1, 1).Address(rowAbsolute:=False)</p> <p>В диалоговом окне отображается адрес \$A1.</p> <p>MsgBox Cells(1, 1).Address(referenceStyle:=xlR1C1)</p> <p>В диалоговом окне отображается адрес R1C1</p>
Cut	<p>Копирует диапазон с удалением в указанный диапазон или в буфер обмена.</p> <p>Синтаксис:</p> <p>Cut(destination)</p> <p>Аргумент destination определяет диапазон, который копируется в данный диапазон. Если аргумент destination опущен, то диапазон копируется в буфер обмена</p>
Columns, Rows	<p>Возвращают соответственно семейства столбцов и строк, из которых состоит диапазон. В следующем примере переменным i и j присваиваются значения, равные количеству столбцов и строк в выделенном диапазоне соответственно:</p> <p>i = Selection.Columns.Count</p> <p>j == Selection.Rows.Count</p>

### Методы объекта Range, использующие команды Excel

Встроенные в Excel команды и методы позволяют эффективно работать с диапазоном: заполнять его элементами по образцу, сортировать, фильтровать и консолидировать данные, строить итоговую таблицу и создавать сценарии, решать нелинейное уравнение с одной переменной.

#### Метод AutoFill

Метод AutoFill (автозаполнение) автоматически заполняет ячейки диапазона элементами последовательности. Метод AutoFill отличается от метода DataSeries тем, что явно указывается диапазон, в котором будет располагаться прогрессия. Вручную этот метод эквивалентен расположению указателя мыши на маркере заполнения выделенного диапазона (в который введены значения, порождающие создаваемую последовательность) и протаскиванию маркера заполнения вдоль диапазона, в котором будет располагаться создаваемая последовательность.

#### Синтаксис:

объект. AutoFill(диапазон, тип)

#### Аргументы:

Диапазон Диапазон, с которого начинается заполнение тип Допустимые значения: xlFillDefault, xlFillSeries, xlFillCopy, xlFillFormats, xlFillValues, xlFillDays, xlFillWeekdays, xlFillMonths, xlFillYears, xlLinearTrend, xlGrowthTrend. По умолчанию xlFillDefault

#### Метод AutoFilter

Метод AutoFilter (автофильтр) представляет собой простой способ запроса и фильтрации данных на рабочем листе. Если AutoFilter активизирован, то каждый заголовок поля выделенного диапазона данных превращается в поле с раскрывающимся списком. Выбирая запрос на вывод данных в поле с раскрывающимся списком, осуществляется вывод только тех записей, которые удовлетворяют указанным условиям. Поле с раскрывающимся списком содержит следующие типы условий: Все (All), Первые десять (Top 10), Условие (Custom), конкретный элемент данных, Пустые (Blanks) и Непустые (NonBlanks). Вручную метод запускается посредством выбора команды Данные, Фильтр, Автофильтр (Data, Filter, AutoFilter).

При применении метода AutoFilter допустимы два синтаксиса.

*Синтаксис 1:*

Объект. AutoFilter

В этом случае метод AutoFilter выбирает или отменяет команду Данные, Фильтр, Автофильтр (Data, Filter, AutoFilter), примененную к диапазону, заданному в аргументе объект.

*Синтаксис 2:*

Объект. AutoFilter (field, criteria1, operator, criteria2)

В этом случае метод AutoFilter выполняет команду Данные, Фильтр, Автофильтр (Data, Filter, AutoFilter) по критериям, указанным в аргументе.

*Аргументы:*

field Целое, указывающее поле, в котором производится фильтрация данных

Criteria1 Задают два возможных условия фильтрации и criteria2 поля. Допускается использование строковой постоянной, например 101, и знаков отношений >, <, >=, <=, =, <>

operator Допустимые значения: X1And (логическое объединение первого и второго критериев); X1or (логическое сложение первого и второго критериев)

При работе с фильтрами полезны метод showAllData и свойства FilterMode и AutoFilterMode.

Метод ShowAllData Показывает все отфильтрованные и неотфильтрованные строки рабочего листа

свойство FilterMode Допустимые значения: True (если на рабочем листе имеются отфильтрованные данные со скрытыми строками), False (в противном случае)

Свойство AutoFilterMode Допустимые значения: True (если на рабочем листе выведены раскрывающиеся списки метода AutoFilter), False (в противном случае)

*Метод GoalSeek*

Метод GoalSeek (подбор параметра) подбирает значение параметра (неизвестной величины), являющееся решением уравнения с одной переменной. Предполагается, что уравнение приведено к виду: правая часть является постоянной, не зависящей от параметра, который входит только в левую часть уравнения. Вручную метод GoalSeek выполняется с помощью команды Сервис, Подбор параметра (Tools, Goal Seek). Метод GoalSeek вычисляет корень, используя метод последовательных приближений, результат выполнения которого, вообще говоря, зависит от начального приближения. Поэтому для корректности нахождения корня надо позаботиться о корректном указании этого начального приближения.

*Синтаксис:*

Объект. GoalSeek(Goal, ChangingCell)

*Аргументы:*

Объект Ячейка, в которую введена формула, являющаяся правой частью решаемого уравнения. В этой формуле роль параметра (неизвестной величины) играет ссылка на ячейку, указанную в аргументе ChangingCell

Goal Значение левой части решаемого уравнения, не содержащей параметра



ChangingCell Ссылка на ячейку, отведенную под параметр (неизвестную величину). Значение, введенное в данную ячейку до активизации метода Goalseek, рассматривается как начальное приближение к искомому корню

Точность, с которой находится корень и предельно допустимое число итераций, используемых для нахождения корня, устанавливается свойствами Maxchange и Maxiterations объекта Application. Например, определение корня с точностью до 0,0001 максимум за 1000 итераций устанавливается инструкцией:

With Application

Maxiterations = 1000

MaxChange = 0.0001

End With

Вручную эти величины устанавливаются на вкладке Вычисления (Calculation) диалогового окна Параметры (Options), вызываемого командой Сервис, Параметры (Tools, Options).

#### *Method Sort*

Сортировка позволяет выстраивать данные в лексикографическом порядке по возрастанию или убыванию. Метод sort осуществляет сортировку строк списков и баз данных, а также столбцов рабочих листов с учетом до трех критериев, по которым производится сортировка. Сортировка данных вручную совершается с использованием команды Данные, Сортировка (Data, Sort).

#### *Синтаксис:*

Объект. Sort(key1, order1, key2, order2, key3, order3, header, orderCustom, matchCase, orientation)

#### *Аргументы:*

Объект Диапазон, который будет сортироваться

Key1 Ссылка на первое упорядочиваемое поле

Order1 Задаёт порядок упорядочивания. Допустимые значения: xlAscending (возрастающий порядок); xlDescending (убывающий порядок)

key2 Ссылка на второе упорядочиваемое поле

order2 Задаёт порядок упорядочивания. Допустимые значения: xlAscending (возрастающий порядок); xlDescending (убывающий порядок)

header Допустимые значения: xlYes (первая строка диапазона содержит заголовок, который не сортируется); xlNo (первая строка диапазона не содержит заголовка, по умолчанию считается данное значение); xlGuess (Excel решает, имеется ли заголовок)

orderCustom Пользовательский порядок сортировки. По умолчанию используется Normal

matchCase Допустимые значения: True (учитываются регистры) и False (регистры не учитываются)

orientation Допустимые значения: xlTopToBottom (сортировка осуществляется сверху вниз, т. е. по строкам); xlLeftToRight (слева направо, т. е. по столбцам)

Например, диапазон A1:C20 рабочего листа лист1 сортируется следующей командой в порядке возрастания так, что первоначальная сортировка происходит по первому столбцу этого диапазона, а второстепенная – по второму:

Worksheets(«Лист»).Range(«A1: C20»).Sort \_

key1:=Worksheets(«Sheet1»).Range(«A1»), \_

key2:=Worksheets («Sheet1»).Range («B1»)

Округление чисел

Округлять десятичные числа приходится часто, особенно при работе с денежными значениями. VBA не предлагает прямого решения таких задач, но обсуждаемые ниже приемы помогут решить эти проблемы.

#### *1 способ*

Функция Round

Пример:

X= round(2.505, 2)

Значение x будет 2,5, а не 2,51.

Поэтому часто не используется.

#### *2 способ*

Функция Format

Пример:

sngОкругление=Format(SngНеокругленное, “#, 0.00”)

#### *3 способ*

Функция FormatNumber

SngОкругление= FormatNumber(sbgНеокругленное, 2)

Для изменения знаков после запятой измените число нулей после десятичной точки в аргументе Format, либо измените число, задающее значение второго аргумента, на нужное.

*Примечание.* Переменная, в которую помещается округленное значение, должна иметь тип string, single, double, decimal, currency или variant, но не тип integer или long.

Приведение данных

Для приведения введенных данных к нужному типу в VBA включен обширный набор функций, одна из которых – CDBL. Синтаксис:

CDBl(выражение)

Обязательный аргумент *выражение* является любым строковым или числовым выражением. Для считывания информации, введенной в текстовое поле в созданной форме, вводят переменную и прописывают выражение:

A = Cdbl(textBoxN.text)

После чего с данной переменной можно работать.

Для вывода значений непосредственно в ячейки книги Excel удобно использовать объект Range:

range(«A5»).value = a

Функцией, обратной по действию к CDBl, является функция CStr – она переводит числа в строки и удобна для вывода результата либо в ячейку на лист, либо в то или иное текстовое окно.

TextBoxN.text = CStr(.Range(«A8»).value)

– считывание значения с ячейки и вывод его в текстовое окно.

Функция Trim (строка) возвращает копию строки, из которой удалены пробелы, находящиеся в начале и конце строки.

Создание VBA-программ

*Использование метода GoalSeek*

*Пример 41.* Разработать программу, которая по введенным числовым значениям некоторого уравнения решает данное уравнение и находит неизвестную переменную x. Результат вычисления выводится в текстовое окно на форме и на лист Excel.

Рис. 92. Разработанная форма примера 41 в рабочем состоянии

Технология выполнения

1. Запустите приложение Excel, сохраните документ.
2. Перейдите в редактор VBA.
3. Создайте форму согласно приведенному рис. 92.
4. На листе Excel расположите необходимый текст (оформление), предусмотрев соответствующие ячейки вывода информации (рис. 93).

	A	B	C
1	<b>Решение уравнения <math>y=a \cdot x^3+b \cdot \sin(x)</math></b>		
2	<b>коэффициенты</b>	<b>значения</b>	
3			
4	<b>a</b>		5
5	<b>b</b>		12
6	<b>y</b>		100
7			99,99999826
8	<b>x</b>		2,663515893
9			

Рис. 93. Вывод результатов на лист excel после запуска формы примера 41

5. Обработайте кнопки.

Кнопка *Вычислить*

```
Private Sub CommandButton1_Click()
```

```
Dim a, b, c As Double
```

```
a = CDbl(TextBox1.Text)
```

```
b = CDbl(TextBox2.Text)
```

```
c = CDbl(TextBox3.Text)
```

```
With ActiveSheet
```

```
Range(«b3»).Value = a
```

```
Range(«b4»).Value = b
```

```
Range(«b5»).Value = c
```

```
Range(«b6»).FormulaLocal = «=b3*b7^3+b4*sin(b7)»
```

```
Range(«b6»).GoalSeek Goal:=c, changingCell:=Range(«b7»)
```

```
TextBox4.Text = CStr(.Range(«b7»).Value)
```

```
TextBox4.Text = FormatNumber(TextBox4.Text, 2)
```

```
End With
```

```
End Sub
```

Кнопка *Заккрыть*

```
Private Sub CommandButton2_Click()
UserForm1.Hide
End Sub
Процедура инициализации формы
Private Sub UserForm_initialize()
Worksheets(1).Visible = False
End Sub
```

Использование методов AutoFill при заполнении таблиц

*Пример 42.* Создать программу, которая по введенным текстовым данным в соответствующие текстовые поля формы автоматизирует ввод данных на студентов некоторой специальности учебного заведения. Результаты заполнения текстовых полей выводятся на лист excel, что позволяет при необходимости распечатать данные.

Технология выполнения

1. Запустите приложение Excel, сохраните документ.
2. Перейдите в редактор VBA. Создайте форму согласно приведенному рис. 94.

Рис. 94. Разработанная форма примера 42 в режиме конструктора

3. На листе Excel расположите необходимый текст (оформление), предусмотрев соответствующие ячейки вывода информации (рис. 95).

	A	B	C	D
1	Список студентов направления 09.03.03			
2				
3	№	Фамилия	группа	адрес
4	1	Иванов	ПИН-15	Ставрополь
5	2	Петров	ПИН16	Ставрополь
6	3	Сидоров	ПИН-13	Ставрополь
7	4	Грецов	ПИН-14	Ставрополь
8				
9				
10	Куртор			Брыкалова А.А.

Рис. 95. Вывод результатов на лист excel после запуска формы

#### 4. Обработайте кнопки.

*Кнопка Создать таблицу*

```
Const strNomer = 3 'количество строк для заголовка
Dim strName1 As String 'строка для адресации ячеек
Dim strName2 As String
Dim nomer As Long 'номер очередной строки таблицы

Private Sub CommandButton1_Click()
ActiveWorkbook.SaveAs ("работа с базой данных. xls")
nomer = 1
End Sub
```

---

*Кнопка Добавить строку*

```
Private Sub CommandButton2_Click()
strName1 = Trim(Str(strNomer + nomer))
With ActiveSheet 'ввод данных для новой отчетной таблицы
Range("A" + strName1).Value = nomer
Range("B" + strName1).Value = TextBox1.Text
Range("C" + strName1).Value = TextBox2.Text
Range("D" + strName1).Value = TextBox3.Text
'автозаполнение с текущей строки таблицы
strName2 = Trim(Str(strNomer + nomer + 1))
Set range1 = .Range("A" + strName1 + ":D" + strName1)
Set range2 = .Range("A" + strName1 + ":D" + strName2)
range1.AutoFill Destination:=range2
Range("A" + strName2 + ":D" + strName2).Clear
End With
' очистка полей формы для ввода очередной записи
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox1.SetFocus
nomer = nomer + 1
End Sub
```

---

*Кнопка Закончить таблицу*

```
Private Sub CommandButton3_Click()

'закрытие формы подведение итогов и вывод фамилии преподавателя
UserForm1.Hide
With ActiveSheet
strName2 = Trim(Str(strNomer + nomer + 2))
Range("A" + strName2).Value = "Куратор"
Range("D" + strName2).Value = TextBox4.Text
End With
End Sub

End Sub
```

#### 5. Откомпилируйте программу и запустите на выполнение.

*Пример 43.* Разработать программу, которая по введенным переменным в соответствующие поля формы решает простейшее линейное уравнение  $y = a \cdot x + b \cdot x$ , находит неизвестную переменную  $x$  и выводит результат вычислений на рабочий лист Excel.

*Пример 44.* Разработать программу, которая по введенным переменным в соответствующие поля формы решает уравнение вида  $y = a \cdot x^3 + 3b \cdot \sin x$ , находит неизвестную переменную  $x$  и выводит результат вычислений на рабочий лист Excel.

*Пример 45.* Разработать программу, которая по введенным переменным в соответствующие поля формы решает уравнение вида  $y = 5a \cdot x^{1/3} + 3b \cdot \lg 4x$ , находит неизвестную переменную  $x$  и выводит результат вычислений на рабочий лист Excel.

*Пример 46.* Разработать программу, которая по введенным переменным в соответствующие поля формы решает уравнение вида  $y = \ln(a \cdot x^3) + 3b \cdot \cos(e^x)$ , находит неизвестную переменную  $x$  и выводит результат вычислений на рабочий лист Excel.

## 7.2. Использование возможностей VBA при непосредственных расчетах

Создание VBA-программ

*Пример 47.* Дан табличный документ указанного ниже вида. Необходимо:

1) создать шаблонную часть этого документа с помощью табличного процессора Excel;

2) составить программу на языке VBA, которая будет запрашивать у пользователя исходные данные для заполнения этой таблицы, производить необходимые расчеты и помещать все данные в соответствующие ячейки, предусмотренные в шаблоне.

Отклонение фактического уровня издержек обращения от плана за месяц 20\_\_ г.

№ п/п	Общество	Сумма издержек		Товарооборот		Уровень издержек, %		Отклонение по уровню, +, -
		план	факт	план	факт	план	факт	
1	2	3	4	5	6	7	8	9
Итого:		*	*	*	*	*	*	

Звездочкой (\*) помечены те графы таблицы, по которым необходимо подвести итог.

### Технология выполнения

Анализ таблицы показывает, что вид деятельности, прогноз прибыли и фактическая прибыль являются исходными данными, отклонение (в процентах и в сумме) – расчетными. Кроме того, рассчитываются итоги по некоторым графам таблицы.

*Создание шаблона табличного документа*

Шаблон создается на обычном рабочем листе в Excel. При этом необходимо только зарезервировать свободные ячейки для занесения следующих данных: месяц, год, потребительское общество, сумма издержек, товарооборот, уровень издержек. Поскольку заранее неизвестно количество потребительских обществ, то ячейки для итогов и ФИО экономиста заранее не резервируются. Рабочий лист переименован в Отчет. Реализация такого шаблона представлена на рис. 96.

	A	B	C	D	E	F	G	H	I
1	<b>Отклонение фактического уровня издержек</b>								
2	<b>обращения от плана</b>								
3				за месяц				года	
4									
5	№ n/n	Общество	Сумма издержек		Товарооборот		Уровень издержек, %		Отклонение по уровню, +,-
6			план.	факт.	план	факт	план	факт	
7	1	2	3	4	5	6	7	8	9
8									
9									
10									
11									
12									
13	Экономист:								

Рис. 96. Шаблон-заготовка табличного документа

На этом рисунке желтым цветом обозначены те ячейки, которые во время работы программы будут заполняться исходными и расчетными данными.

#### *Математическая модель решения задачи*

Кроме организации ввода исходных данных и вывода их в некоторые ячейки электронной таблицы, программа должна производить расчет отклонений и итоговых значений по графам «Сумма издержек – план», «Сумма издержек – факт», «Товарооборот – план», «Товарооборот – факт», «Уровень издержек – план», «Уровень издержек – факт», «Отклонение по уровню». Для расчетных величин используем следующие переменные:

Номер – номер текущей строки таблицы;

SP – планируемая сумма издержек;

SF – фактическая сумма издержек;

TP – планируемый товарооборот;

TF – фактический товарооборот;

IP – планируемый уровень издержек;

EF – фактический уровень издержек;

ItogSP – накопление итога по столбцу «планируемая сумма издержек»;

ItogSF – накопление итога по столбцу «фактическая сумма издержек»;

ItogTP – накопление итога по столбцу «планируемый товарооборот»;

ItogTF – накопление итога по столбцу «фактический товарооборот»;

ItogIP – накопление итога по столбцу «планируемый уровень издержек»;

ItogEF – накопление итога по столбцу «фактический уровень издержек».

С учетом введенных обозначений расчетные формулы будут иметь следующий вид:

1) для отклонений:

$$[\text{Отклонение в \%}] = (F - P) / P * 100$$

$$[\text{Отклонение в сумме}] = F - P$$

Результаты этих вычислений можно не сохранять в отдельных переменных, так как они сразу могут быть занесены в соответствующие ячейки электронной таблицы;

2) для итогов по прогнозу и факту:

$$\text{ItogP} = \text{ItogP} + P$$

$$\text{ItogF} = \text{ItogF} + F$$

Эти формулы реализуют алгоритм получения итоговой суммы методом накопления, когда величина прогноза (факта), соответствующая очередному виду деятельности, добавляется к сумме соответствующих величин по уже рассмотренным видам деятельности. Назовем эти суммы промежуточными. Когда будут обработаны все виды деятельности, промежуточные суммы превратятся в окончательные – итоговые. В начале этого процесса (до того, как будет рассчитываться первая промежуточная сумма) переменные ItogP и ItogF равны нулю;

3) для итогов по отклонениям:

[итоговое отклонение в процентах] =  $(ItogF - ItogP) / ItogP * 100$

[итоговое отклонение в сумме] =  $ItogF - ItogP$

Результаты этих вычислений можно не сохранять в отдельных переменных, так как они сразу могут быть занесены в соответствующие ячейки электронной таблицы.

#### *Разработка интерфейса пользователя*

Каждому текстовому полю поменяем стандартное имя (TextBox) на более понятное (рис. 97). В нашем примере:

TextBox1 – MesTextBox – ввод месяца;

TextBox2 – YearTextBox – ввод года;

TextBox3 – FIOTextBox – ввод фамилии, имени и отчества экономиста;

TextBox4 – POTextBox – ввод названия потребительского общества;

TextBox5 – SPTextBox – планируемая сумма издержек;

TextBox6 – SFTextBox – фактическая сумма издержек;

TextBox7 – TPTextBox – планируемый товарооборот;

TextBox8 – TFTextBox – фактический товарооборот;

TextBox9 – IPTTextBox – планируемый уровень издержек;

TextBox10 – EFTTextBox – фактический уровень издержек.

Рис. 97. Разработанная форма примера 47 в рабочем состоянии



```

' Объявление переменных и констант
Const StrNomer = 7 'количество строк для заголовка таблицы
Dim Nomer As Long 'номер очередной строки таблицы (потребительское общество)
Dim SP As Long
Dim SF As Long
Dim TP As Long
Dim TF As Long
Dim IP As Long
Dim EF As Long
Dim ItogSP As Long
Dim ItogSF As Long
Dim ItogTP As Long
Dim ItogTF As Long
Dim ItogIP As Long
Dim ItogEF As Long
Dim StrName1 As String
Dim StrName2 As String

```

```

'Процедура инициализации формы
Private Sub UserForm_Initialize()
Worksheets(«Отчет»).Activate
MesTextBox.SetFocus
End Sub

```

```

'Процедура считывания заголовочных данных и вывода их в ячейки электронной
таблицы Private Sub CommandButton3_Click()
'Ввод данных для новой отчетной таблицы
With ActiveSheet
Range(«E3»).Value = MesTextBox.Text
Range(«G3»).Value = YearTextBox.Text
End With
ActiveWorkbook.SaveAs ("Отклонение фактического уровня издержек обращения от
плана за " + MesTextBox.Text + «месяц. xls»)
Nomer = 1
ItogSP = 0
ItogSF = 0
ItogTP = 0
ItogTF = 0
ItogIP = 0
ItogEF = 0
End Sub

```

```

' Процедура обработки данных по видам деятельности
Private Sub CommandButton2_Click()
StrName1 = Trim(Str(StrNomer + Nomer))
With ActiveSheet
Range("A" + StrName1).Value = Nomer
Range("B" + StrName1).Value = POTextBox.Text
SP = Val(SPTextBox.Text)
Range("C" + StrName1).Value = SP
ItogSP = ItogSP + SP
SF = Val(SFTextBox.Text)

```

```

Range("D" + StrName1).Value = SF
ItogSF = ItogSF + SF
TP = Val(TPTextBox.Text)
Range("E" + StrName1).Value = TP
ItogTP = ItogTP + TP
TF = Val(TFTextBox.Text)
Range("F" + StrName1).Value = TF
ItogTF = ItogTF + TF
IP = Val(IPTextBox.Text)
Range("G" + StrName1).Value = IP
ItogIP = ItogIP + IP
EF = Val(EFTextBox.Text)
Range("H" + StrName1).Value = EF
ItogEF = ItogEF + EF
Range("I" + StrName1).Value = ItogEF - ItogIP
'Выполнение автозаполнения с текущей строки таблицы на следующую строку
StrName2 = Trim(Str(StrNomer + Nomer + 1))
Set Range1 = Range("A" + StrName1 + ":I" + StrName1)
Set Range2 = Range("A" + StrName1 + ":I" + StrName2)
Range1.AutoFill Destination:=Range2
Range("A" + StrName2 + ":I" + StrName2).ClearContents
End With

```

```

'Очистка полей формы для ввода очередных данных
POTextBox.Text = ""
SPTextBox.Text = ""
SFTextBox.Text = ""
TPTextBox.Text = ""
TFTextBox.Text = ""
IPTextBox.Text = ""
EFTextBox.Text = ""
POTextBox.SetFocus
Nomer = Nomer + 1
End Sub

```

```

'Закрытие формы, подведение итогов и вывод фамилии экономиста
Private Sub CommandButton1_Click()
UserForm1.Hide
StrName1 = Trim(Str(StrNomer + Nomer))
With ActiveSheet
Range("A" + StrName1).Value = «Итого:»
Range("C" + StrName1).Value = ItogSP
Range("D" + StrName1).Value = ItogSF
Range("E" + StrName1).Value = ItogTP
Range("F" + StrName1).Value = ItogTF
Range("G" + StrName1).Value = ItogIP
Range("H" + StrName1).Value = ItogEF
Range("I" + StrName1).Value = ItogEF - ItogIP
StrName2 = Trim(Str(StrNomer + Nomer + 2))
Range("A" + StrName2).Value = «Экономист:»
Range("G" + StrName2).Value = FIOTTextBox.Text
End With

```

End Sub

*Использование программы в режиме выполнения и получения результатов*

Перед запуском программы сохраним рабочую книгу под именем отчет1.xls.

Кнопку «Создать отчетную таблицу» следует нажимать один раз после ввода заголовочной информации, кнопка «Добавить строку» нажимается каждый раз после ввода данных по очередному виду деятельности. После ввода всех данных необходимо нажать кнопку «Закончить», а затем переключиться в окно Microsoft Excel. На рабочем листе появится результат, аналогичный рис. 98.

Задачи на закрепление материала

*Пример 48.* Создать программу, которая по введенным в соответствующие текстовые поля формы данным автоматизирует ввод данных по обслуживанию населения некоторой организацией за определенный период времени. Кроме этого, программа должна вычислять данные по столбцам, отмеченным звездочками. Вывод данных предусмотреть на лист Excel.

	A	B	C	D	E	F	G	H	I
1	<b>Отклонение фактического уровня издержек</b>								
2	<b>обращения от плана</b>								
3				за месяц	май		2006	года	
4									
5	№ п/п	Общество	Сумма издержек		Товаро-оборот		Уровень издержек, %		Откло- нение по уровню, +,-
6			план.	факт.	план	факт	план	факт	
7	1	2	3	4	5	6	7	8	9
8	1	ОАО "Ликсар"	200	150	250	300	150	165	15
9	2	ОАО "Балтекс"	450	350	1500	2000	100	50	-50
10	3	ОАО "Прицеп"	2500	2400	45000	50000	45	50	5
11	Итого:		3150	2900	46750	52300	295	265	-30

Рис. 98. Шаблон табличного документа, заполненный данными

### Численность обслуживаемого населения ..... в 200.... году

№	Наименование района	Мужчины	Женщины	Дети	В том числе дошкольники	Итого
1	2	3	4	5	6	7
Всего по области:		*	*	*	*	*

Экономист

В.И. Петров

*Пример 49.* Создать программу, которая по введенным в соответствующие текстовые поля формы данным автоматизирует ввод данных по расчету товарного баланса некоторой организацией, занимающейся торговлей, за определенный период времени. Кроме этого, программа должна вычислять данные по столбцам, отмеченным звездочками. Вывод данных предусмотреть на лист excel.

**Товарный баланс**  
..... за ... кв. 200... г. (тыс. руб.)

№	Товарная группа	Запас на начало периода	Поступило	Продано	Запас на конец периода
1	2	3	4	5	6
Итого:		*	*	*	*

Начальник планового отдела

В.И. Семенов

*Пример 50.* Создать программу, которая по введенным в соответствующие текстовые поля формы данным автоматизирует ввод данных на отпуск товара с некоторого склада и формирует счет-фактуру за определенный период времени. Кроме этого, программа должна вычислять данные по столбцам, отмеченным звездочками. Вывод данных предусмотреть на лист excel.

**Счет-фактура**  
на отпуск товара со склада № ..... межрайбазы

№	Наименование и артикул товара	Код	Ед. измер.	Количество	Цена (руб.)	Сумма (тыс. руб.)
1	2	3	4	5	6	7
Итого:						*

Зав. складом  
Гл. бухгалтер

К.И. Иванов  
Г.А. Петров

*Пример 51.* Создать программу, которая по введенным в соответствующие текстовые поля формы данным автоматизирует ввод данных на студентов определенной специальности. Данные выводятся на рабочий лист excel. Предусмотреть на листе excel кнопку, которая вызывает необходимую форму для пользователя, не загружая редактор vba.

Технология выполнения

1. Сохраните новый документ Excel.
2. Создайте форму приложения, отвечающую требованиям задачи (см. рис. 99а).
3. Пропишите процедуры обработки нажатия кнопок *Создать отчетную таблицу*, *Добавить строку* и *Закончить* (см. пример 47).
4. При выводе информации на рабочий лист Excel придерживайтесь оформления, предложенного на рис. 99б.

**Специальность АСОИиУ**

Заголовок таблицы

курс  группа

куратор

Данные для таблицы

ФИО

год рождения

адрес по прописке

адрес в Балашове

отец

мать

Рис. 99а. Разработанная форма примера 51 в рабочем состоянии

	A	B	C	D	E	F	G
1	<b>Список студентов</b>						
2	<b>специальности АСОИ и У</b>						
3	курс	4		группа		АС-42	
4							
5	№	ФИО	год рождения	адрес по прописке	адрес в Балашове	отец	мать
6	1	Белянский Виктор	1986	с.Красавка	Ул.Нижняя,25	Белянский В.В.	Белянская Л.Н.
7	2	Бобылева Надежда	1985	с.Пески	общежитие	Бобылев Н.В.	Бобылева Н.Ю.
8	3	Гладкова Наталья	1986	с. Пески	общежитие	Гладков В.Д.	Гладкова С.В.
9							
10							
11		классный руководитель		Фризен И.Г.			

Рис. 99б. Вывод информации на лист excel после нажатия кнопки «Закончить»

5. Произведем презентацию формы, т. е. создадим кнопку запуска формы непосредственно на листе Excel, так как запускать форму на выполнение каждый раз из редактора VBA не очень красиво и неудобно для пользователя. Для простоты использования формы поступим следующим образом: установим на рабочем листе Excel со списком группы кнопку «Диалог», нажатие которой будет приводить к появлению разработанной формы (диалогового окна). Для этого:

1) выберите в главном меню Microsoft Excel <Разработчики> <Макрос><Макросы...>;

2) укажите имя макроса, например Макрос1, и нажмите <Создать>. После этого загрузится редактор VBA с заготовкой

```
Sub Макрос1()
```

```
End Sub
```

3) вставьте в эту заготовку оператор UserForm1.Show, активизирующий созданную форму;

4) далее выберите в главном меню книги Excel <Вид><Панели инструментов><Формы>. Появится панель с элементами управления. Выберите <Кнопка>, после чего можно нарисовать кнопку на рабочем листе Excel;

5) после этого сразу появляется диалоговое окно «Назначить макрос объекту». Выберите Макрос1;

6) чтобы название кнопки было более содержательным, щелкните на ней правой клавишей мыши. В появившемся рядом меню выберите <Изменить текст>. После этого установите текст «Диалог».

Теперь при нажатии кнопки «Диалог» на рабочем листе появится разработанное диалоговое окно примера.

### 7.3. Финансовые функции

Расчет амортизации

VBA предоставляет ряд встроенных функций, которые можно использовать для выполнения финансовых расчетов. Они разбиты на три основные группы: функции учета амортизации, функции учета отчислений и функции учета денежных потоков. Мы рассмотрим одну из этих групп – функции учета амортизации. Эти функции используются в бухгалтерском деле для предоставления в денежном выражении обесценивания основных средств за определенный период времени. Например, фирме, являющейся владельцем грузового автомобиля, необходимо рассчитать сумму ежегодной амортизации грузовика для вычисления текущей стоимости грузовика в любой момент времени. Поскольку амортизация влияет на размер налогов, правительство часто устанавливает обязательные формулы, которые следует применять для расчета амортизации.

Под амортизацией понимают уменьшение стоимости имущества (основных средств) в процессе эксплуатации. Обычно оценивают величину этого уменьшения, приходящуюся на единицу времени. В финансовый пакет VBA включены три функции для расчетов амортизации (табл. 21).

Таблица 21

Общие параметры функций для расчетов амортизации

Параметр	Значение
cost (стоимость)	Начальная стоимость имущества
salvage (остаток)	Остаточная (ликвидная) стоимость имущества
life (время_экспл)	Длительность периода эксплуатации
period (период)	Номер периода для вычисления амортизации

Функция SLN вычисляет амортизацию (снижение стоимости) за один период времени, используя метод равномерной амортизации. Ее вызов имеет вид:

$SLN(cost, salvage, life) = SLN(стоимость, остаток, период)$

Пусть компьютер стоимостью 6000 руб. имеет срок эксплуатации 5 лет, после чего его стоимость оценивается в 1500 руб. Тогда его ежегодная амортизация составит (при равномерном методе оценки):  $SLN(6000, 1500, 5) = 900$  руб.

Функция SYD используется для расчета годовой амортизации по линейному методу. Сумма долей амортизации в этом методе определяется как сумма номеров лет эксплуатации:

$$1+2+\dots+life = life(life+1)/2,$$

а доля амортизации за *i*-й год пропорциональна количеству лет (*life* – *i* + 1), оставшихся до конца периода эксплуатации. Синтаксис этой функции:

$$SYD(cost, salvage, life, period) = SYD(\text{стоимость}, \text{остаток}, \text{время\_экспл}, \text{период})$$

При расчете предыдущего примера получим за первый год эксплуатации компьютера амортизацию

$$SYD(6000, 1500, 5, 1) = 1500 \text{ руб.}$$

а за последний год —

$$SYD(6000, 1500, 5, 5) = 300 \text{ руб.}$$

*Примечание.* Все параметры указываются через запятую.

Функция DDB вычисляет величину амортизации имущества для заданного периода с применением метода двукратного (или *k*-кратного) учета амортизации. В этом методе амортизация максимальна в первый период и снижается в последующие периоды.

Синтаксис:

$$DDB(cost, salvage, life, period, factor) = DDB(\text{нач\_стоим}, \text{остаток}, \text{время\_экспл}, \text{период}, \text{коэффициент})$$

Параметр *factor* (*коэффициент*) – это норма снижения балансовой стоимости (амортизации). По умолчанию он равен 2 (метод двукратного учета амортизации).

Функция DDB использует следующую формулу для вычисления амортизации за период:

$$\frac{\text{Стоимость} - \text{Остаточная стоимость} - \text{Суммарная амортизация за предшествующие периоды}}{\text{Время эксплуатации}} \times \text{Коэффициент.}$$

Если нужно использовать другой метод вычисления амортизации, измените значение коэффициента.

В примере с компьютером по методу двукратной амортизации она составит:

$$\text{за первый год: } DDB(6000, 1500, 5, 1) = 2400 \text{ руб.};$$

$$\text{за второй} - 1440 \text{ руб.};$$

$$\text{за третий} - 660 \text{ руб.};$$

$$\text{а за четвертый и пятый будет равна } 0.$$

Все пять аргументов должны быть положительными числами.

*Замечания.* Метод двойного процента со снижающегося остатка вычисляет амортизацию, используя увеличенный коэффициент. Амортизация максимальна в первый период, в последующие периоды уменьшается.

*Примеры.* Предположим, что предприятие приобрело новую машину. Машина стоит 2400 \$ и имеет срок службы 10 лет. Остаточная стоимость составляет 300 \$. Следующие примеры показывают амортизацию за несколько периодов. Результаты округляются до двух знаков после запятой.

$$ddb(2400, 300, 3650, 1) \text{ равняется } 1,32 \$, \text{ амортизация за первый день.}$$

$$\text{Microsoft Excel автоматически предполагает, что коэффициент равен } 2.$$

$$ddb(2400, 300, 120, 1) \text{ равняется } 40,00 \$, \text{ амортизация за первый месяц.}$$

$$ddb(2400, 300, 10, 1) \text{ равняется } 480,00 \$, \text{ амортизация за первый год.}$$

$$ddb(2400, 300, 10, 2, 1,5) \text{ равняется } 306,00 \$, \text{ амортизация за второй год при использовании коэффициента, равного } 1,5 \text{ (а не метода двойного процента).}$$

$$ddb(2400, 300, 10, 10) \text{ равняется } 22,12 \$, \text{ амортизация за десятый год. Microsoft Excel автоматически предполагает, что коэффициент равен } 2.$$

### Создание VBA-программы

*Пример 52.* Создать программу, которая по введенным данным в текстовые поля формы рассчитывает амортизацию имущества за заданный период времени с использованием метода двукратного учета. Диалоговое окно расчета амортизации должно предусматривать: ввод исходных данных (начальная и остаточная стоимости, срок эксплуатации в годах), задание параметров амортизации (день, месяц, год и какой по счету).

Вычисление амортизации по заданным условиям и вывод отчета произвести на рабочий лист excel. При этом следует предусмотреть, чтобы кнопка «Вывести отчет» являлась недоступной до вычисления амортизации.

Технология выполнения

1. Запустите приложение Excel и сохраните книгу под соответствующим именем.
2. Создайте форму аналогично приведенному рис. 100.
3. Пропишите работу кнопки «Вычислить»:

```
Private Sub CommandButton1_Click()
```

```
Dim timeall As Integer
```

Рис. 100. Разработанная форма примера 52 в рабочем состоянии

```
Dim amort As Double
```

```
With ComboBox1
```

```
If.ListIndex = 0 Then
```

```
timeall = TextBox3.Value * 365
```

```
ElseIf.ListIndex = 1 Then
```

```
timeall = TextBox3.Value * 12
```

```
Else
```

```
timeall = TextBox3.Value
```

```
End If
```

```
End With
```

```
amort = DDB(TextBox1.Value, TextBox2.Value, timeall,  
TextBox4.Value)
```

```
TextBox5.Value = amort
```

```
CommandButton2.Enabled = True
```

```
End Sub
```

Обратите внимание на вычисление значений параметра timeall в зависимости от выбора пункта в раскрывающемся списке (день, месяц, год) и использования функции DDB.

4. Заполнение поля со списком происходит в процедуре инициализации формы:

```
Private Sub UserForm_initialize()
```

```
With ComboBox1
```

```
AddItem «день»
```

```
AddItem «месяц»
```

```
AddItem «год»
```

```
ListIndex = 0
```

```
End With
```

```
End Sub
```



Эта процедура запускается при использовании известного вам оператора Show.

5. Нажатие кнопки «Вывести отчет» должно приводить к выводу отчета на рабочий лист Excel в следующем формате (рис. 101):

	A	B	C	D	E	F
1	начальная стоимость фонда				15000	
2	остаточная стоимость фонда				14000	
3	длительность периода эксплуатации				10	
4	период времени для амортизации				5	
5						
6	амортизация составила				24.7233796296259	месяц
7						

Рис. 101. Вывод вычислений на лист excel в примере 52

Задача на закрепление материала

*Пример 53.* Функция ddb() имеет пять аргументов, последний из которых является коэффициентом амортизации. Необходимо в форме предусмотреть новое текстовое поле, куда пользователь смог бы вводить коэффициент, а в коде нужно учесть присутствие этого коэффициента.

В программе необходимо построить обработчик ошибок. Необходимость его создания обусловлена следующим фактором: если весь период эксплуатации составляет, к примеру, 120 месяцев, то невозможно вычислить амортизацию за 121 и т. д. месяцы. Таким образом, при некорректном вводе периода расчета амортизации должно появляться окно вывода с сообщением об ошибке и должен быть осуществлен новый ввод неверного параметра.

*Указание:* выход из процедуры осуществляется командой Exit Sub.

## Лабораторная работа №8 Построение диаграмм средствами VBA

### 8.1. Построение гладких диаграмм

#### Диаграммы в Excel

Диаграммы можно размещать на рабочем листе. Для этого используется коллекция `chartobjects`. Ее элементы – объекты класса `chartobject` – являются контейнерами, содержащими объект `Chart`, задающий непосредственно диаграмму.

#### Методы объекта *Chart*

Рассмотрим основные методы, определяющие новое поведение объекта `Chart`:

*SubChartWizard* ([Source], [Gallery], [Format], [PlotBy], [CategoryLabels], [SeriesLabels], [HasLegend], [Title], [CategoryTitle], [ValueTitle], [Extra-Title])

Этот метод позволяет построить или модифицировать существующую диаграмму. В отличие от мастера диаграмм (`ChartWizard`), который вызывается при построении диаграммы вручную, метод не является интерактивным, более того, он не позволяет задать все возможные свойства. С его помощью можно выполнить основную работу, а детали строятся с использованием других свойств и методов объекта `Chart`. Все параметры метода являются необязательными:

1) *Source* – объект `Range`, содержащий исходные данные для построения новой диаграммы. Если параметр опущен, то метод позволяет отредактировать существующую диаграмму – либо выделенную диаграмму рабочего листа, либо диаграмму активного листа диаграмм;

2) *Gallery* – задает тип диаграммы и может быть одной из следующих констант: `xlArea`, `xlBar`, `xlColumn`, `xlLine`, `xlPie`, `xlRadar`, `xlXY-Scatter`, `xlCombination`, `xl3DArea`, `xl3DBar`, `xl3DColumn`, `xl3DLine`, `xl3DPie`, `xl3DSurface`, `xlDoughnut`, или `xlDefaultAutoFormat`;

3) *Format* – задает формат для данного типа диаграммы. Каждому типу диаграммы соответствует некоторое число возможных форматов. Параметр задает номер формата, по умолчанию выбирается первый формат данного типа;

4) *PlotBy* – соответствует терминальному свойству `PlotBy`;

5) *CategoryLabels* и *SeriesLabels* – целые, указывающие число строк или столбцов с метками категорий и рядов данных в области, заданной параметром `Source`. Указывать эти числа нужно на единицу меньше фактического значения;

6) остальные параметры позволяют добавить легенду, задать название диаграммы и ее осей – они совпадают с соответствующими терминальными свойствами.

*Sub SetSourceData* (source as range, [plotby]). Устанавливает источник данных диаграммы. Второй параметр соответствует терминальному свойству `plotby`.

*Sub ApplyCustomType* (ChartTypeAsXlChartType, [typename]). Метод позволяет модифицировать диаграмму, применив к ней новый тип – стандартный или настраиваемый. Если этот тип стандартный, то тогда первый параметр полностью его определяет. Его возможные значения совпадают со значениями соответствующего терминального свойства `charttype`. Если же тип настраиваемый, то первый параметр должен иметь одно из следующих значений: `xlbuiltin`, `xluserdefined` или `xlanygallery`. В этом случае второй параметр задает имя типа диаграммы.

*Function Export*(filename as string, [filtername], [interactive]) as boolean позволяет экспортировать диаграмму, преобразуя ее в графический формат. Первый параметр задает имя файла, в который будет записана диаграмма в графическом формате, второй – задает имя графического фильтра в том виде, как оно записано в системном регистре. Булев параметр `interactive` должен иметь значение `true`, если мы хотим вызвать диалоговое окно в процессе фильтрации. Функция `export` возвращает значение `true` в случае успешного завершения работы.

*Sub GetChartElement*(x as long, y as long, elementid as long, arg1 as long, arg2 as long). Представьте себе, что пользователь щелкнул кнопку мыши где-то над диаграммой. Обработав это событие, можно получить координаты курсора мыши – `x` и `y`. Если теперь вызвать метод `getchartelement` с этими координатами, то он вернет значение параметра

elementid – идентификатор элемента диаграммы и значения двух параметров, связанных с этим элементом. Конечно, параметры зависят от типа того элемента, чьи координаты  $x$  и  $y$  заданы.

*Function Location* (where as xlchartlocation, [name]) as

Chart. Передвигает диаграмму в новое местоположение. Параметр Where имеет следующие значения: xlLocationAsNewSheet, xlLocationAsObject или xlLocationAutomatic.

В первом случае диаграмма помещается на новый лист диаграммы и параметр Name задает имя этого листа. Во втором случае диаграмма помещается как встроенный объект и Name задает имя рабочего листа.

Создание VBA-программы

*Пример 54.[6]* По введенным в диалоговое окно «Построение графика» (рис. 102) начальным, конечным значениям аргументов и их шагам изменения строится график. Уравнение графика также вводится в программу из диалогового окна. Уравнение должно быть составлено в соответствии с правилами, по которым строятся функции рабочего листа, но в качестве аргументов в нем следует использовать  $x$  вместо ссылки на ячейку. Программа сама переведет аргумент в ссылку на ячейку. После табуляции введенной функции программой и построения поверхности на рабочем листе (рис. 103) этот график также отображается в объекте управления Image, расположенном в диалоговом окне *Построение графика* (рис. 102).

Технология выполнения

Обсудим, как приведенная ниже программа решает описанную задачу и что происходит в ней.

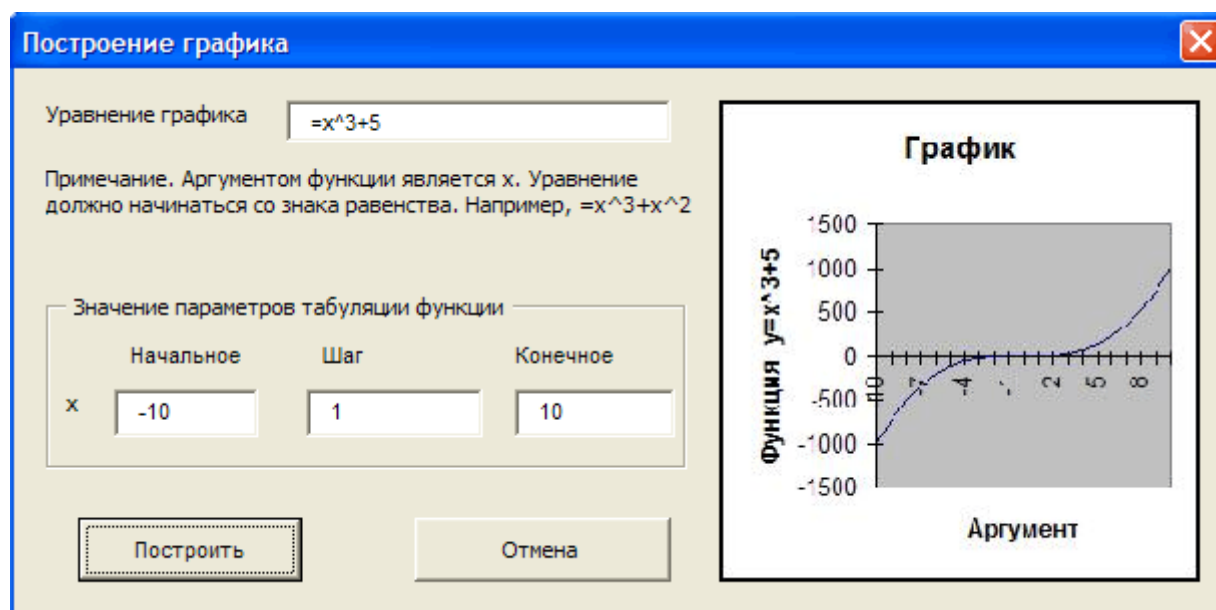


Рис. 102. Диалоговое окно «Построение графика» в рабочем состоянии (пример 54)

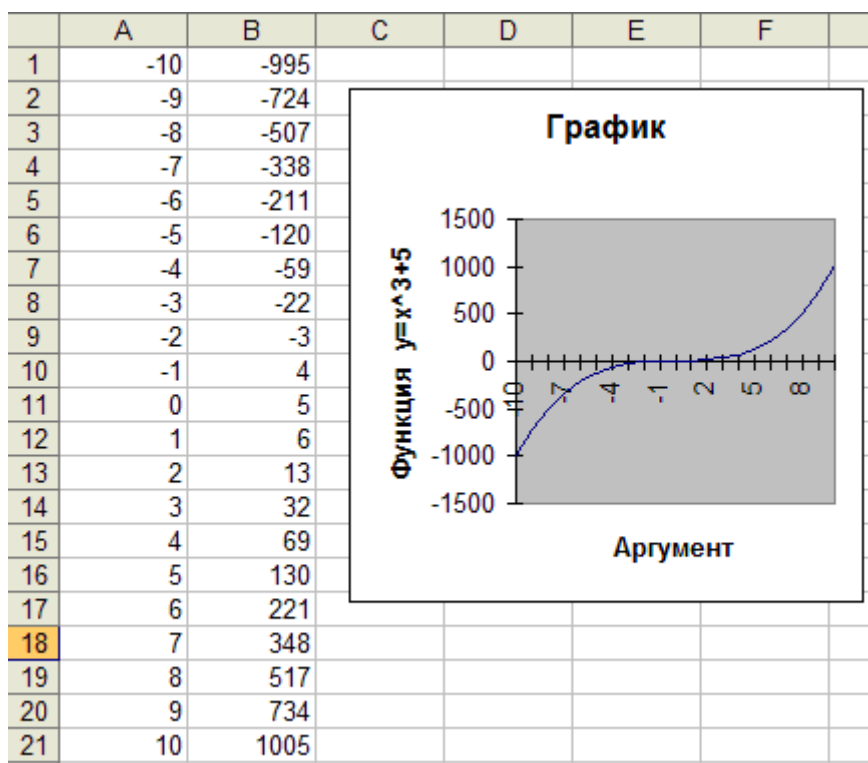


Рис. 103. Результат построения графика на рабочем листе (пример 54)

#### *UserForm Initialize*

1. Активизирует диалоговое окно.
2. Назначает клавише <Esc> функцию кнопки *Отмена*, а клавише <Enter> – построение.
3. Устанавливает, чтобы отображаемая картинка графика в диалоговом окне помещалась целиком и пропорционально в пределах элемента управления Image, а также чтобы левый верхний угол рисунка совпадал с левым верхним углом элемента управления Image.

*Нажатие кнопки Построение запускает на выполнение процедуру CommandButton1\_Click*

1. Проверяет, являются ли вводимые данные числами. В случае ошибки отображается соответствующее сообщение.
2. Проверяет согласованность вводимых данных. В случае ошибки отображается соответствующее сообщение (рис. 104).
3. Преобразует формулу, введенную в поле Уравнение графика, в формулу рабочего листа.
4. Проверяет корректность введенной формулы. В случае ошибки отображается соответствующее сообщение (рис. 105).
5. Используя метод *DataSeries*, начиная с ячейки A2, строит вниз по столбцу арифметическую прогрессию, являющуюся результатом табуляции аргумента  $x$  уравнения графика с указанными шагами (рисунок из файла *graph.jpg* в элементе управления Image1).

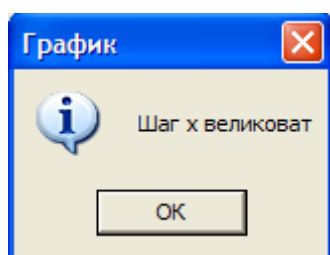


Рис. 104. Пример сообщения о несогласованности данных

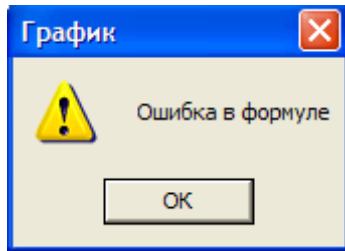


Рис. 105. Сообщение о некорректном вводе формулы  
*Нажатие кнопки Отмена запускает на выполнение процедуру*  
*CommandButton2\_Click*

Закрывает диалоговое окно.

Рассмотрим листинг данного приложения.

Private Sub CommandButton1\_Click()

*' Процедура табуляции функции*

Dim x\_нз As Double

Dim x\_пз As Double

Dim x\_шаг As Double

Dim УрГрафика As String

Dim nx As Integer

*'nx – число протабулированных значений аргумента x*

Dim n As Integer

Dim i As Integer

*'n,i – вспомогательные целые переменные*

*'Проверка корректности ввода данных*

If IsNumeric(TextBox2.Text) = False Then

MsgBox «Ошибка в начальном значении x», vbInformation, «График»

TextBox2.SetFocus

Exit Sub

End If

If IsNumeric(TextBox3.Text) = False Then

MsgBox «Ошибка в шаге x», vbInformation, «График»

TextBox3.SetFocus

Exit Sub

End If

If IsNumeric(TextBox4.Text) = False Then

MsgBox «Ошибка в конечном значении y», vbInformation, «График»

TextBox4.SetFocus

Exit Sub

End If

*'Считывание с диалогового окна значений переменных*

x\_нз = CDBl(TextBox2.Text)

x\_шаг = CDBl(TextBox3.Text)

x\_пз = CDBl(TextBox4.Text)

УрГрафика = Trim(TextBox1.Text)

*'Проверка согласованности введенных данных*

If x\_нз >= x\_пз Then

MsgBox «Начальное значение x слишком большое», vbInformation, «График»

TextBox2.SetFocus

Exit Sub

End If

If x\_нз + x\_шаг >= x\_пз Then

```

MsgBox «Шаг х великоват», vbInformation, «График»
TextBox3.SetFocus
Exit Sub
End If
'Замена в введенной формуле аргумента х на ссылку $A1
i = 1
Do
'Замена в введенной формуле аргумента х на ссылку $A1
If Mid(УрГрафика, i, 1) = «х» Or Mid(УрГрафика, i, 1) = «X» Then
n = Len(УрГрафика)
If (1 < i) And (i < n) Then
УрГрафика = Left(УрГрафика, i - 1) & «$A1» & Right(УрГрафика, n - i)
End If
If i = 1 Then
УрГрафика = «$A1» & Right(УрГрафика, n - 1)
End If
If i = n Then
УрГрафика = Left(УрГрафика, n - 1) & «$A1»
End If
End If
i = i + 1
Loop While i <= Len(УрГрафика)
ActiveSheet.Cells.Select
Selection.Clear
'Очистка на активном листе ранее введенных данных
ActiveSheet.Range(«A1»).Select
'Заполнение диапазонов значениями аргумента
With ActiveSheet
Range(«A1»).Value = x_нз 'Ввод в ячейку A1 начального значения
'Создание арифметической прогрессии по столбцу с указанным шагом и начальным
значением
Range(«A1»).DataSeries Rowcol:=xlColumns, Type:=xlLinear, Step:=x_шаг,
Stop:=x_пз, Trend:=False
End With
'Заполнение диапазона значениями функции
With ActiveSheet
nx = Range(«A1»).CurrentRegion.Rows.Count
'Определение числа строк в диапазоне заполнения
Range(«B1»).FormulaLocal = УрГрафика
'Ввод уравнения поверхности в ячейку B1
If IsError(Evaluate(УрГрафика)) = True Then
MsgBox «Ошибка в формуле», vbExclamation, «График»
Exit Sub
End If
'Заполнение диапазона Range(Cells(1, 2), Cells(nx, 2))
'начиная с ячейки B1, что эквивалентно протаскиванию маркера
'заполнения ячейки B1 на диапазон Range(Cells(1, 2),
Cells(nx, 2))
Range(«B1»).AutoFill Destination:=Range(Cells(1, 2), Cells(nx, 2)), Type:=xlFillDefault
End With
ActiveSheet.ChartObjects.Delete
'Удаление с рабочего листа всех ранее построенных диаграмм

```

```

ActiveSheet.Range(Cells(1, 2), Cells(nx, 2)).Select
'Выбор диапазона, по которому строится график
ActiveSheet.ChartObjects.Add(20, 19.5, 192, 192).Select
'Задание и выбор области на рабочем листе, где будет построен график,
'размер графика должен соответствовать размеру объекта Image1
Application.CutCopyMode = False
'Построение графика
ActiveChart.ChartWizard Source:=Range(Cells(1, 1), Cells(nx, 2)), Gallery:=xlLine,
Format:=2, PlotBy:=xlColumns, CategoryLabels:=1, SeriesLabels:=0, HasLegend:=False,
Title:="График",
CategoryTitle:="Аргумент", ValueTitle:="Функция y" & TextBox1.Text
ActiveSheet.ChartObjects(1).Activate
ActiveChart.Axes(xlValue).AxisTitle.Select
With Selection
HorizontalAlignment = xlCenter
VerticalAlignment = xlCenter
Orientation = xlUpward
End With
'Запись диаграммы в файл и загрузка картинки в Image1
ActiveChart.Export Filename:="Graph.jpg",
FilterName:="JPEG"
UserForm1.Image1.Picture = LoadPicture(«graph.jpg»)
ActiveSheet.Range(«A1»).Select
End Sub
Private Sub CommandButton2 Click()
'Процедура закрытия диалогового окна
UserForm1.Hide
End Sub
Private Sub UserForm Initialize()
'Рисунок масштабируется с учетом относительных размеров
так, чтобы он помещался в объекте Image1
With Image1
PictureAlignment = fmPictureAlignmentTopLeft
PictureSizeMode = fmPictureSizeModeStretch
End With
End Sub

```

## 8.2. Построение круговых диаграмм и гистограмм

Создание VBA-программ

*Пример 55.* В диалоговое окно «Построение графика» (рис. 104) вводятся начальное, конечное значения аргумента и его шаг изменения. Уравнение графика жестко регламентировано. Программа сама переведет аргумент в ссылку на ячейку. После табуляции введенной функции программой и построения поверхности на рабочем листе этот график также отображается в объекте управления image, расположенном в диалоговом окне *Построение графика* (рис. 106, 107).

*Пример 56.* В диалоговое окно «Построение графика» (рис. 108) вводятся начальное, конечное значения аргумента и его шаг изменения. Уравнение графика вводится в текстовое окно. Программа сама переведет аргумент в ссылку на ячейку. Предусматривается выбор типа графика при построении графика функции. После табуляции введенной функции программой и построения поверхности на рабочем листе этот график также отображается в объекте управления image, расположенном в диалоговом окне *Построение графика* (рис. 108, 109, 110).

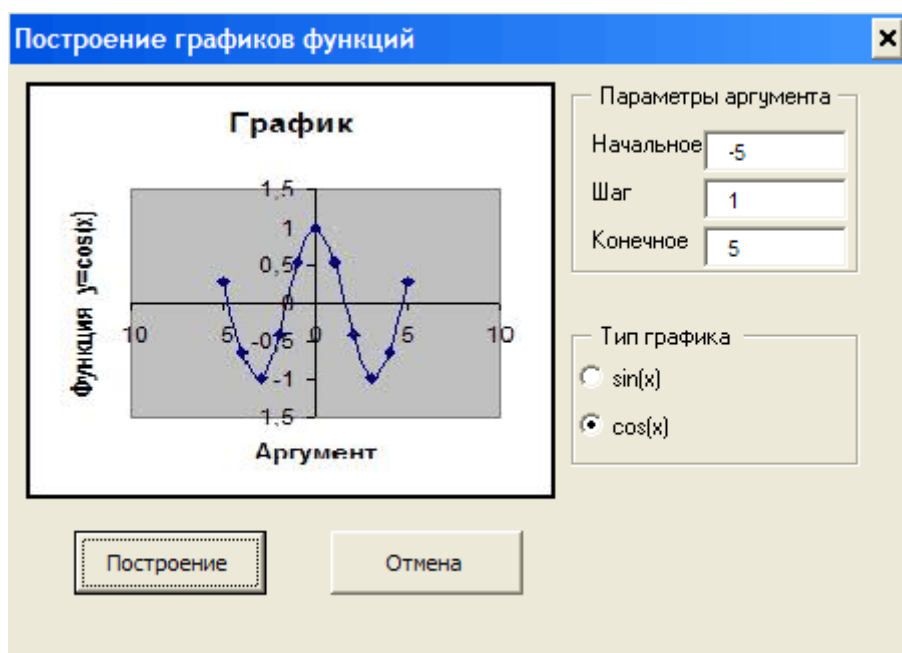


Рис. 106. Построение графика в диалоговом окне при выборе  $y = \cos(x)$

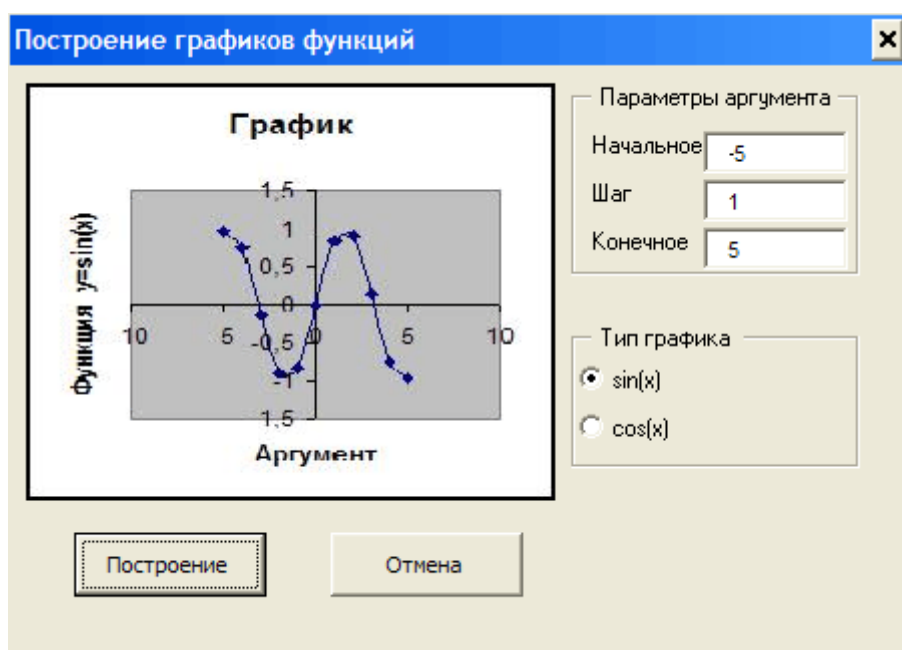


Рис. 107. Построение графика в диалоговом окне при выборе  $y = \sin(x)$



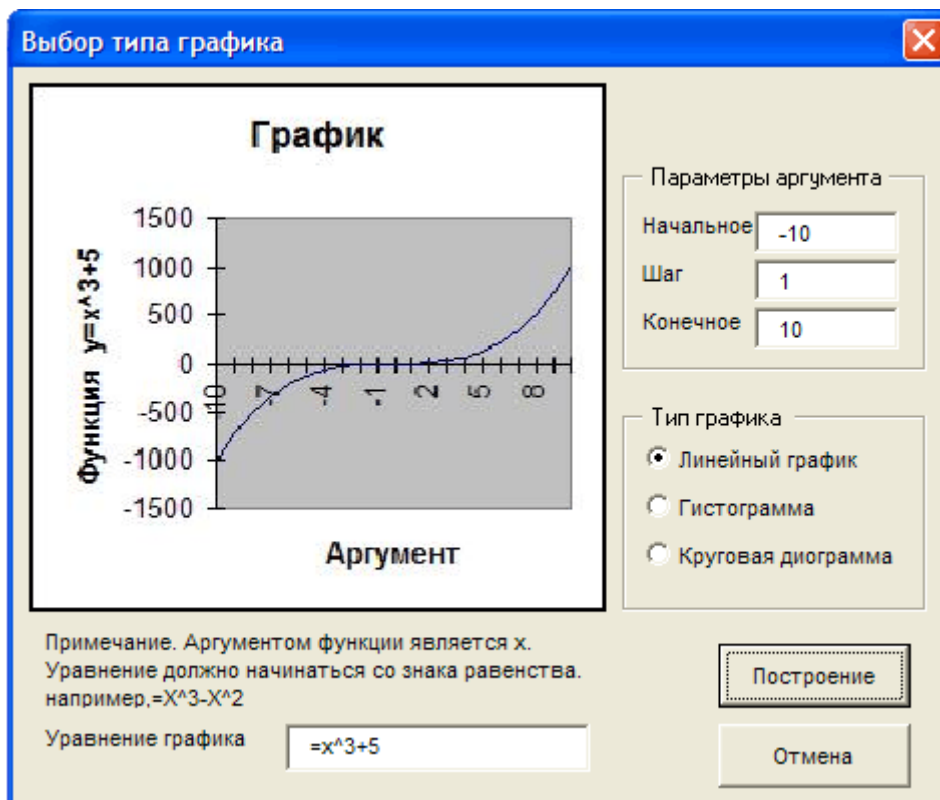


Рис. 108. Построение графика в диалоговом окне при выборе линейного типа

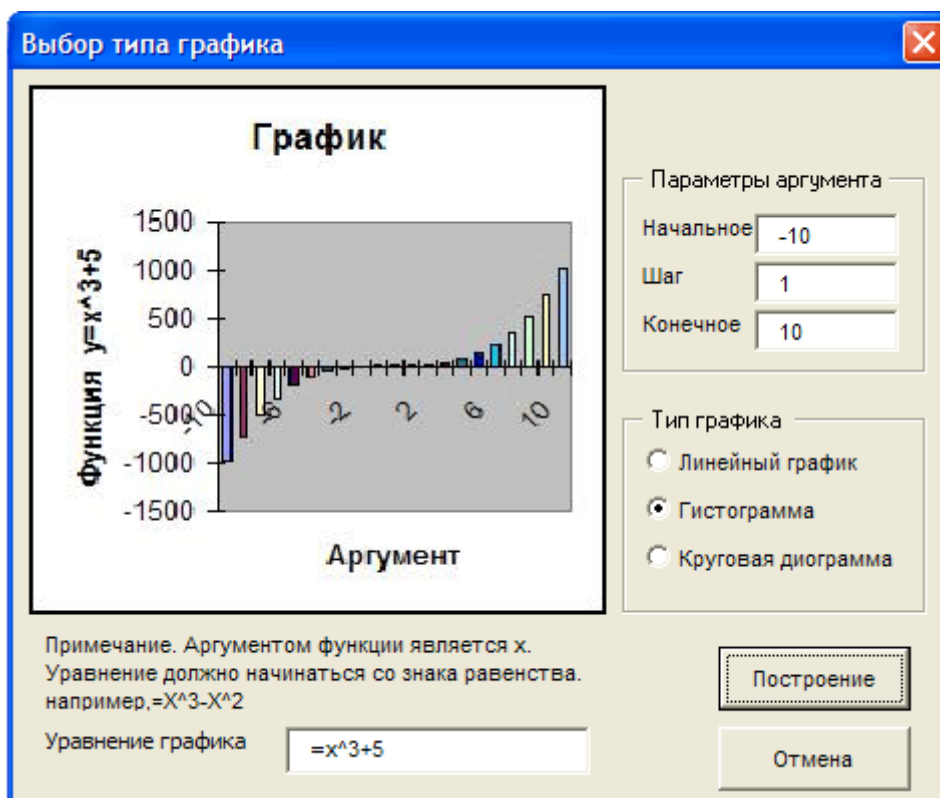


Рис. 109. Построение графика в диалоговом окне при выборе гистограммы

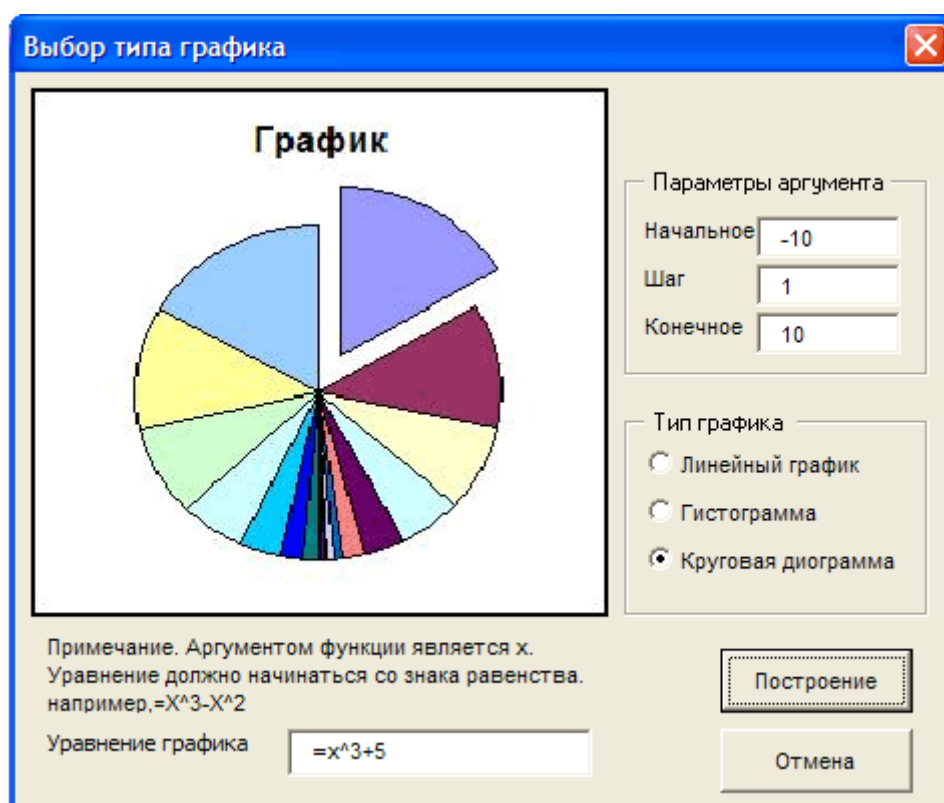


Рис. 110. Построение графика в диалоговом окне при выборе круговой диаграммы

## Лабораторная работа №9. Базы данных в Excel

### 9.1. Заполнение базы данных

#### Элементы управления

При разработке приложений, кроме рассмотренных ранее элементов управления, можно использовать элемент «полоса прокрутки» и «счетчик». Рассмотрим эти элементы.

#### Элемент управления *ScrollBar*



(полоса прокрутки) и элемент управления *SpinButton*



(счетчик) по своим функциональным возможностям аналогичны друг другу. Поэтому свойства их рассматриваем совместно.

*Value* – возвращает текущее значение полосы прокрутки (только целые неотрицательные числа);

*Min* – минимальное значение полосы прокрутки (только целые неотрицательные числа);

*Max* – максимальное значение полосы прокрутки (только целые неотрицательные числа);

*SmallChange* – устанавливает шаг изменения значения при щелчке по одной из стрелок полосы прокрутки;

*Enabled* – допустимые значения: True (пользователь может изменить значение полосы) и False (в противном случае);

*Visible* – допустимые значения: True (полоса прокрутки отображается во время выполнения программы) и False (в противном случае).

#### Создание VBA-программы

При разработке формы будет необходимо выполнять программирование примечаний ячеек листа. Данная процедура выполняется с помощью метода *AddComment* объекта *Range*. Этот метод имеет следующий синтаксис:

*AddComment* (Text),

где *Text* – текст комментария.

*Пример 57.[7]* Для заполнения базы данных на рабочем листе excel с помощью редактора пользовательских форм создать диалоговое окно *Регистрация клиентов* (рис. 111).

С помощью этого диалогового окна вводятся данные в базу (таблицу базы), расположенную на рабочем листе Excel (рис. 112). Нажатие кнопки ОК должно приводить к добавлению новой записи в таблицу.

Рис. 111. Вид рабочей формы примера 57

	A	B	C	D	E	F	G	H
1	Фамилия	Имя	Пол	Выбранный Тур	Оплачено	Фото	Паспорт	Срок
2	Светлякова	Светлана	Жен	Лондон	Да	Да	Да	3
3	Орлов	Сергей	Муж	Париж	Да	Да	Да	5
4	Павленко	Александр	Муж	Берлин	Да	Нет	Да	7
5	Рудаков	Дмитрий	Муж	Берлин	Да	Нет	Нет	7
6	Иванова	Светлана	Жен	Берлин	Да	Да	Да	6

Рис. 112. База данных о туристах на рабочем листе

Технология выполнения

Обсудим, как приведенная ниже программа решает перечисленные задачи и что происходит в программе.

*UserForm\_Initialize*

1. Активизирует диалоговое окно.
2. Назначает клавише <Esc> функцию кнопки Отмена, а клавише <Enter> –

Вычислить.

3. Назначает кнопкам *Вычислить*, *Отмена* и переключателям всплывающие подсказки.

4. Закрепляет первую строку так, чтобы она всегда отображалась на экране.

5. Создает заголовки полей базы данных, если они еще не были созданы.

6. Устанавливает начальное значение переключателя «0 программе».

7. Заполняет раскрывающийся список.

8. Устанавливает текст заголовка окна приложения.

*CommandButton1\_Click*

1. Определяет номер первой пустой строки в базе данных о регистрации туристов, куда будет введена новая запись.

2. Считывает данные из диалогового окна.

3. Вводит их в первую пустую строку.

*CommandButton2\_Click*

1. Закрывает диалоговое окно.

2. Устанавливает заголовок приложения, используемый по умолчанию, т. е. удаляет пользовательский заголовок приложения, созданный при активизации формы.

*SpinButton1\_Change*

Вводит значение в поле Продолжительность тура.

Private Sub CommandButton1\_Click()

' Процедура считывания информации из диалогового окна

' и записи ее в базу данных на рабочем листе

' Смысл переменных однозначно определен их названиями

Dim Фамилия As String \* 20

Dim Имя As String \* 20

Dim Пол As String \* 3

Dim ВыбранныйТур As String \* 20

Dim Оплачено As String \* 3

Dim Фото As String \* 3

Dim Паспорт As String \* 3

Dim Срок As String \* 3

Dim НомерСтроки As Integer

'НомерСтроки – номер первой пустой строки рабочего листа НомерСтроки = Application.CountA(ActiveSheet.Columns(1)) + 1

'Считывание информации из диалогового окна в переменные With UserForm1

Фамилия = TextBox1.Text

```

Имя = .TextBox2.Text
Срок = .TextBox3.Text
Пол = If(.OptionButton1.Value, «Муж», «Жен»)
Оплачено = If(.CheckBox1.Value, «Да», «Нет»)
Фото = If(.CheckBox2.Value, «Да», «Нет»)
Паспорт = If(.CheckBox3.Value, «Да», «Нет»)
ВыбранныйТип = .ComboBox1.List(.ComboBox1.ListIndex, 0)
End With
'Ввод данных в строку с номером НомерСтроки рабочего листа
With ActiveSheet
Cells(НомерСтроки, 1).Value = Фамилия
Cells(НомерСтроки, 2).Value = Имя
Cells(НомерСтроки, 3).Value = Пол
Cells(НомерСтроки, 4).Value = ВыбранныйТип
Cells(НомерСтроки, 5).Value = Оплачено
Cells(НомерСтроки, 6).Value = Фото
Cells(НомерСтроки, 7).Value = Паспорт
Cells(НомерСтроки, 8).Value = Срок
End With
End Sub

```

В приведенной процедуре для определения первой пустой строки в заполняемой базе данных о туристах используется инструкция

```
НомерСтроки = Application.CountA(ActiveSheet.Columns(1)) + 1,
```

правая часть которой вычисляет число непустых ячеек в первом столбце активного рабочего листа. Переменной *НомерСтроки* присваивается значение на единицу большее, чем число непустых строк, что естественно, так как ей должен быть присвоен номер первой непустой строки базы данных. Подобные инструкции довольно часто используются при разработке приложений, поэтому следует обратить на них внимание.

```

Private Sub SpinButton1_Change()
'Процедура ввода значения счетчика в поле ввода
With UserForm1
TextBox3.Text = CStr(.SpinButton1.Value)
End With
End Sub
Private Sub TextBox3_Change()
'Процедура установки значения счетчика из поля ввода
With UserForm1
SpinButton1.Value = CInt(.TextBox3.Text)
End With
End Sub

```

```

Private Sub CommandButton2_Click()
'Процедура закрытия диалогового окна
UserForm1.Hide
Application.Caption = Empty
'Установка заголовка окна приложения по умолчанию
End Sub

```

```

Private Sub UserForm_Initialize()
'Процедура вызова диалогового окна
'и задание элементов раскрывающегося списка
'Задание пользовательского заголовка окна приложения

```

```

Application.Caption = «Регистрация. База данных туристов фирмы 'Балашов-Тур'»
Application.DisplayFormulaBar = False 'Заккрытие строки формул окна Excel
With CommandButton1
Default = True
ControlTipText = «Ввод данных в базу данных»
End With
With CommandButton2
Cancel = True
ControlTipText = «Кнопка отмены»
End With
OptionButton1.Value = True
With ComboBox1
'Задание элементов раскрывающегося списка
List = Array(«Лондон», «Париж», «Берлин»)
ListIndex = 0
End With
'Задание начального и минимального значений счетчика и вывод текста
SpinButton1.Value = 1
SpinButton1.Min = 1
ЗаголовокРабочегоЛиста
UserForm1.Show
'Активизация диалогового окна
End Sub

```

```

Sub ЗаголовокРабочегоЛиста()
'Процедура создания заголовков полей базы данных
If Range(«A1»).Value = «Фамилия» Then Range(«A2»).Select Exit Sub 'Если заголовки
существуют, то досрочный выход из процедуры
End If 'Если заголовки не существуют, то создаются заголовки полей
ActiveSheet.Cells.Clear
Range(«A1:H1»).Value = Array(«Фамилия», «Имя», «Пол», «Выбранный Тур»,
«Оплачено», «Фото», «Паспорт», «Срок»)
Range(«A: A»).ColumnWidth = 12
Range(«D: D»).ColumnWidth = 14.4
Range(«2:2»).Select
'Закрепляется первая строка с тем, чтобы она всегда отображалась на экране
ActiveWindow.FreezePanels = True
Range(«A2»).Select
'К каждому заголовку поля базы данных присоединяется примечание
Range(«A1»).AddComment
Range(«A1»).Comment.Visible = False
Range(«A1»).Comment.Text Text:="Фамилия клиента"
Range(«B1»).AddComment
Range(«B1»).Comment.Visible = False
Range(«B1»).Comment.Text Text:="Имя клиента"
Range(«C1»).AddComment
Range(«C1»).Comment.Visible = False
Range(«C1»).Comment.Text Text:="Пол клиента"
Range(«D1»).AddComment
Range(«D1»).Comment.Visible = False
Range(«D1»).Comment.Text Text:="Направление" & Chr(10) & «выбранного тура»
Range(«E1»).AddComment

```

```

Range(«E1»).Comment.Visible = False
Range(«E1»).Comment.Text Text:="Путевка оплачена?" & Chr(10) & «(Да/Нет)»
Range(«F1»).AddComment
Range(«F1»).Comment.Visible = False
Range(«F1»).Comment.Text Text:="Фото сданы" & Chr(10) & «(Да/Нет)»
Range(«G1»).AddComment
Range(«G1»).Comment.Visible = False
Range(«G1»).Comment.Text Text:="Наличие паспорта" & Chr(10) & "(Да/Нет)»
Range(«H1»).AddComment
Range(«H1»).Comment.Visible = False
Range(«H1»).Comment.Text Text:="Продолжительность" & Chr(10) & «поездки»
End Sub

```

При написании процедура *ЗаголовокРабочегоЛиста* лучше всего воспользоваться MacroRecorder, который переведет производимые действия по созданию примечаний пользователем вручную на язык VBA.

Задача на закрепление материала

*Пример 58.* Модифицировать форму примера 57 и, соответственно, базу на рабочем листе (ввести новую колонку, заголовок и комментарий к нему) для хранения еще одного параметра – *Постоянный клиент* (это дает, например, скидку при оплате).

## 9.2. Конструирование пользовательского интерфейса

Создание VBA-программы

Microsoft Excel содержит встроенные средства по созданию и управлению базами данных. Это:

- создание таблицы базы данных (осуществляется при заполнении заголовков полей таблицы);
- заполнение таблицы базы данных (меню <Данные> <Форма>);
- сортировка записей таблицы (меню <Данные> <Сортировка>);
- фильтрация записей таблицы по определенному признаку или группе признаков (меню <Данные> <Фильтр>) и др.

С помощью этих средств осуществляется управление базой в Excel. Данный факт весьма облегчает задачу автоматизации использования баз в Excel. Сводится это к возможности применения макрорекордера для программной реализации базы в Excel с помощью VBA.

*Пример 59.* Создать приложение с пользовательским интерфейсом по заполнению и обработке базы данных туристической фирмы «Балашов-Тур». База данных состоит из двух рабочих листов: «База Данных» и «Фильтр».

После загрузки программа сама будет создавать свой интерфейс, отображать название окна приложения и, если на рабочем листе нет заголовков полей, создавать их. Интерфейс программы будет состоять из нескольких диалоговых окон.

Технология выполнения

Первое диалоговое окно уже существует (пример 57), оно реализует заполнение базы данных. С помощью второго диалогового окна будет реализовываться сортировка записей таблицы.

*Примечание.* Пересохраните работу *Пример57* как *Пример59*.

Второе диалоговое окно (UserForm2, рис. 113) позволяет осуществлять сортировку записей таблицы (рис. 114) по одному из двух критериев:

- продолжительности тура;
- фамилии.

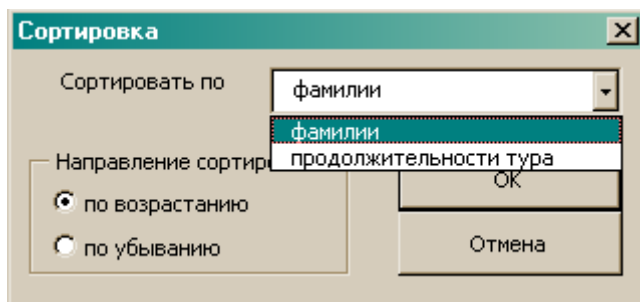


Рис. 113. Форма 2 для примера 59

Также предоставляется выбор сортировки по возрастанию или по убыванию. Интересной особенностью этого диалогового окна является название второй кнопки. При появлении на экран кнопка носит название «Отмена», а после осуществления сортировки получает название «Заккрыть».

Для облегчения написания кода сортировки следует воспользоваться макрорекордером. После включения записи выполните следующие шаги:

- 1) выделите записи базы данных;
- 2) выберите в меню <Данные> <Сортировка>;
- 3) при появлении диалогового окна выберите поле, по которому осуществляется сортировка, а также ее направление;
- 4) нажмите Enter;
- 5) выключите макрорекордер.

При просмотре полученного макроса можно обнаружить несколько операторов, которые станут шаблоном для кода. Это выделение области сортировки (записей базы) и собственно сама процедура сортировки. Примерно так:

```
Range(«A2:H5»).Select
```

```
Selection.Sort Key1:=Range(«A2»), Order1:=xlAscending, Key2:=Range(«B2»),  
Order2:=xlAscending, Key3:=Range(«E2»), Order3:=xlDescending, Header:=xlGuess,  
OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
```

Что придется корректировать:

1) область сортировки. Необходимость корректирования вполне ясна: количество записей может быть меньше либо больше, чем в данный момент. Для определения числа записей воспользуемся известным приемом (см. пример 57);

2) поле, по которому осуществляется сортировка. Необходимо предусмотреть выбор поля перед самой сортировкой (в элементе управления «Поле со списком» (ComboBox1));

3) направление сортировки. Также нужно предусмотреть выбор одного варианта из двух (переключатель OptionButton1 или OptionButton2).

Создайте форму UserForm2. Затем в окне ее кода создайте процедуру для кнопки ОК.

```
Private Sub CommandButton1_Click()
```

```
КоличествоСтрок =
```

```
Application.CountA(ActiveSheet.Columns(1))
```

```
'Количество записей в базе
```

```
Range(Cells(2, 1), Cells(КоличествоСтрок, 8)).Select
```

```
'выделение области сортировки
```

```
If ComboBox1.Value = «фамилии» Then
```

```
KeySort = «A2»
```

```
'ключ сортировки – поле с фамилией
```

```
Else
```

```
KeySort = «H2»
```

```
'ключ сортировки – поле со сроком поездки
```

```
End If
```



```

'Сортировка
If OptionButton1.Value Then
'по возрастанию
Selection.Sort Key1:=Range(KeySort), Order1:=xlAscending, Header:=xlGuess,
OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
Else
'по убыванию
Selection.Sort Key1:=Range(KeySort),
Order1:=xlDescending,Header:=xlGuess, OrderCustom:=1, MatchCase:=False,
Orientation:=xlTopToBottom
End If
Range(«A2»).Select
'установка активной ячейки с первой фамилией
CommandButton2.Caption = «Заккрыть»
'изменение названия второй кнопки
End Sub
Вторая кнопка закрывает форму и возвращает свое исходное имя.
Private Sub CommandButton2_Click()
CommandButton2.Caption = «Отмена»
UserForm2.Hide
End Sub

```

Для инициализации формы UserForm2 откройте Модуль1 и вставьте процедуру инициализации формы.

```

Public Sub UserForm2_Initialize()
' обратите внимание, что процедура глобальная!
UserForm2.ComboBox1.List = Array(«фамилии», «продолжительности тура»)
UserForm2.ComboBox1.ListIndex = 0
UserForm2.Show
End Sub

```

Сортировка Фильтрация								
	A	B	C	D	E	F	G	H
1	Фамилия	Имя	Пол	Выбранный Тур	Оплачено	Фото	Паспорт	Срок
2	Иванченко	Юрий	Муж	Париж	Да	Да	Да	30
3	Руденко	Дмитрий	Муж	Берлин	Да	Нет	Нет	7
4	Поворин	Александр	Муж	Берлин	Да	Нет	Да	7
5	Ленинко	Равин	Муж	Париж	Да	Да	Да	6
6	Ковалева	Светлана	Жен	Лондон	Да	Да	Да	3
7	Горбачев	Николай	Муж	Лондон	Нет	Нет	Нет	1
8	Валиуллин	Антон	Муж	Лондон	Нет	Нет	Нет	1
9	Антипов	Дмитрий	Муж	Лондон	нет	нет	нет	10

Рис. 114. Вывод данных на лист excel

#### Конструирование интерфейса. Презентация

Для создания пользовательского интерфейса следует выполнить следующее:

- 1) установить новое название приложения «Туристы фирмы Балашов-Тур»;
- 2) закрыть строку формул;
- 3) убрать панели инструментов *Стандартная* и *Форматирование*;
- 4) дать новое имя листу с базой;
- 5) добавить новую панель инструментов с кнопкой «Сортировка».

Новое имя листу задайте вручную. Остальное будет сделано в процедуре, обрабатывающей событие открытия рабочей книги.

```

Private Sub Workbook_Open()
Application.Caption = «Туристы фирмы Балашов-тур»
Application.DisplayFormulaBar = False

```

```

'Заккрытие строки формул окна Excel
Application.CommandBars(«Standard»).Visible = False
Application.CommandBars(«Formatting»).Visible = False
Sheets(«База данных»).Select
With Application.CommandBars.Add(Name:="Рабочая панель
инструментов",Position:=msoBarTop, MenuBar:=False, Temporary:=True)
    Visible = True
    With.Controls
        'кнопка Сортировка
        With.Add(Type:=msoControlButton, ID:=1)
        Caption = «Сортировка»
        ToolTipText = «Сортировка»
        Style = msoButtonCaption
        OnAction = «Module1.UserForm2_Initialize»
        'кнопка запускает UserForm2_Initialize
    End With
End With
End With
UserForm1.Show
End Sub

```

Для возвращения внешнего вида приложения после закрытия базы следует обработать событие «Непосредственно перед закрытием».

```

Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Application.Caption = Empty
    'Установка заголовка окна приложения по умолчанию
    Application.CommandBars(«Standard»).Visible = True
    Application.CommandBars(«Formatting»).Visible = True
End Sub

```

Теперь сохраните все изменения и закройте Excel. Далее при открытии файла с базой вы увидите новый интерфейс приложения и подгруженную форму с регистрацией клиента.

Задача на закрепление материала

*Пример 60.[8]* Создать базу данных «Общежитие».

Для этого разработать приложение с диалоговым окном «Общежитие», в котором:

- счетчик управляет вводом продолжительности проживания;
- в раскрывающемся списке выводятся три типа номеров: одноместный, двухместный, люкс, стоимость проживания в которых равна соответственно 550, 400 и 750 руб. в сутки;
- если постоялец заказывает завтраки в номер, то суточная оплата возрастает на 75 руб.;
- при нажатии на кнопку ОК в поле *Стоимость* проживания выводится суммарная стоимость проживания клиента, и все данные из диалогового окна должны выводиться в базу данных, создаваемую на рабочем листе; кроме того, происходит автоматическое сохранение рабочей книги на диск.

При разработке формы придерживаться рис. 115, 116.

Отель "Общежитие БКТ"

Фамилия

Имя

Пол  
☐ Муж  
☒ Жен

Завтрак, документы  
☒ Паспорт  
☒ Завтрак в номер

Номера

Продолжительность проживания

Оплата  
☒ Наличными  
☐ Кредитная карточка  
☐ Бартер

Рис. 115. Форма примера 60 в рабочем состоянии

	A	B	C	D	E	F	G	H	I
1	Фамилия	имя	Пол	Номер	Оплата	Паспорт	Завтра	Срок	Стоимость
2	Иванченко	Павел	Муж	Одноместный	Наличные	Да	Да	4	2500
3	Коваленко	светлана	Жен	Двухместный	Кредитная карт	Нет	Да	6	2850
4	Орлова	Ирина	Жен	Одноместный	Наличные	Да	Да	5	3125
5	Светланова	Ольга	Жен	Люкс	Наличные	Да	Да	1	825
6	Приветов	Петр	Жен	Люкс	Кредитная карт	Нет	Да	3	2475
7									

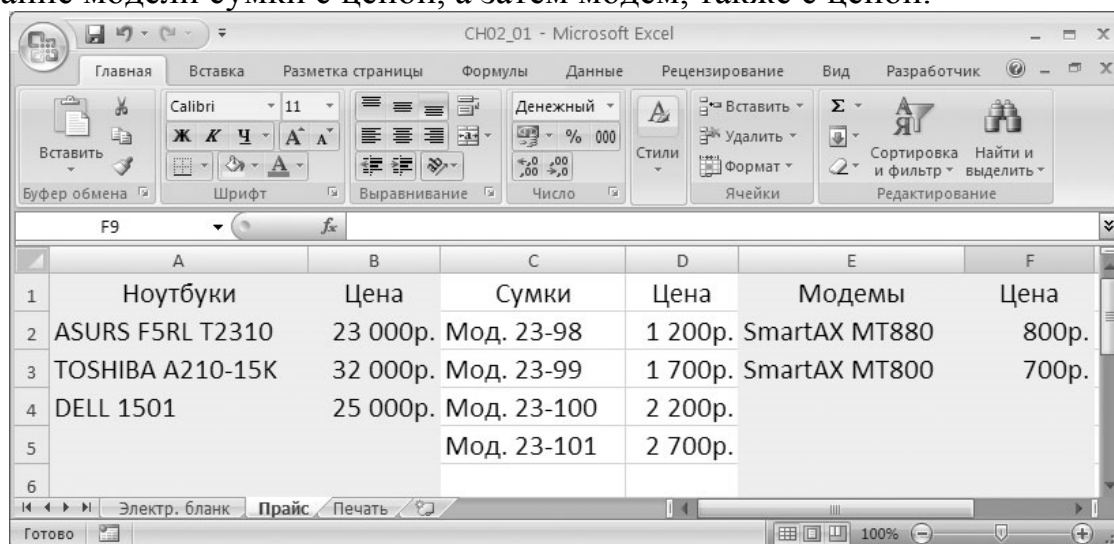
Рис. 116. Вывод информации в базу данных excel

## Лабораторная работа № 10 Создание бланк заказа для обслуживания покупателей

Теперь, после того как мы познакомились с простыми конструкциями доступа из процедур VBA к ячейкам листа Microsoft Excel, пора перейти к более сложной задаче. Будем считать, что нам необходимо разработать удобный бланк заказа при обслуживании покупателей в компьютерном салоне. Наша задача — обеспечить удобный подбор необходимой покупки из возможных вариантов по информации, присутствующей в прайс-листе.

Будем считать, что комплектация состоит максимум из 3 компонентов: ноутбука, сумки к нему и модема. При этом различных моделей ноутбуков, сумок и модемов достаточно много, и эта номенклатура отражается на втором листе книги. Такой лист с названием *Прайс* представлен на рис. 117. Понятно, что строк в прайс-листе может быть достаточно много, и при подборе заказа не очень удобно его постоянно прокручивать.

Наша задача далее заключается в разработке удобного электронного бланка. Заметим только, что в разрабатываемой книге Microsoft Excel предполагается, что прайс лист должен находиться на втором рабочем листе в последовательности листов (в дальнейшем программном коде используется номер листа в качестве параметра семейства Worksheets). На рис. 117 видно, что первая строка прайс-листа представляет собой совокупность заголовков, а сами данные располагаются со второй строки. В столбцах А и В содержится описание модели ноутбука и его цена соответственно. Следующие два столбца — название модели сумки с ценой, а затем модем, также с ценой.



	А	В	С	Д	Е	Ф
1	Ноутбуки	Цена	Сумки	Цена	Модемы	Цена
2	ASURS F5RL T2310	23 000р.	Мод. 23-98	1 200р.	SmartAX MT880	800р.
3	TOSHIBA A210-15K	32 000р.	Мод. 23-99	1 700р.	SmartAX MT800	700р.
4	DELL 1501	25 000р.	Мод. 23-100	2 200р.		
5			Мод. 23-101	2 700р.		
6						

Рис. 117. Содержание листа Прайс

Мы попробуем на первом рабочем листе создать удобный для пользователя электронный бланк заказа, который будет использовать информацию из прайс-листа.

Интерфейс первого листа, который нам необходимо разработать, показан на рис. 118. Разберем технические действия для его создания. Во-первых, необходимо расширить столбцы А, В и С, а затем в ячейку В2 ввести текст «Бланк заказа» и отформатировать его соответствующим образом. После

этого разместим в ячейке B3 другой текст — «Заказ №». Рядом с этим текстом на листе расположим элемент управления «Поле» (или как мы раньше его называли — текстовое окно) для набора с клавиатуры номера заказа. Для свойства Name этого элемента выберем значение Nomer, добавим серый фон окна (с помощью свойства BackColor), а для свойства AutoSize установим в качестве значения True. В этом случае ширина текстового окна будет автоматически изменяться при вводе символов.

Теперь следует заполнить текстом (см. рис. 118) содержимое ячеек B5, C5, A7, A8, A9, B11 и выполнить соответствующее форматирование. После этого уберем с экрана сетку, а для прямоугольного диапазона A5-C14 установим внешние границы. Кроме этого, можно также установить заливку для диапазона A5-C14 (это выполняется с помощью несложных технических действий в Microsoft Excel).

Теперь следует разместить три элемента управления типа «Поле со списком» — с ними мы уже работали в первой главе. Для того чтобы не было противоречий с программным кодом, имена этих элементов выберем (сверху вниз по рис. 118) Spk1, Spk2 и Spk3 соответственно. Таким образом, мы создали интерфейс для нашей разработки. Далее наступает этап программирования созданных элементов, поэтому перейдем в редактор Visual Basic для написания программного кода.

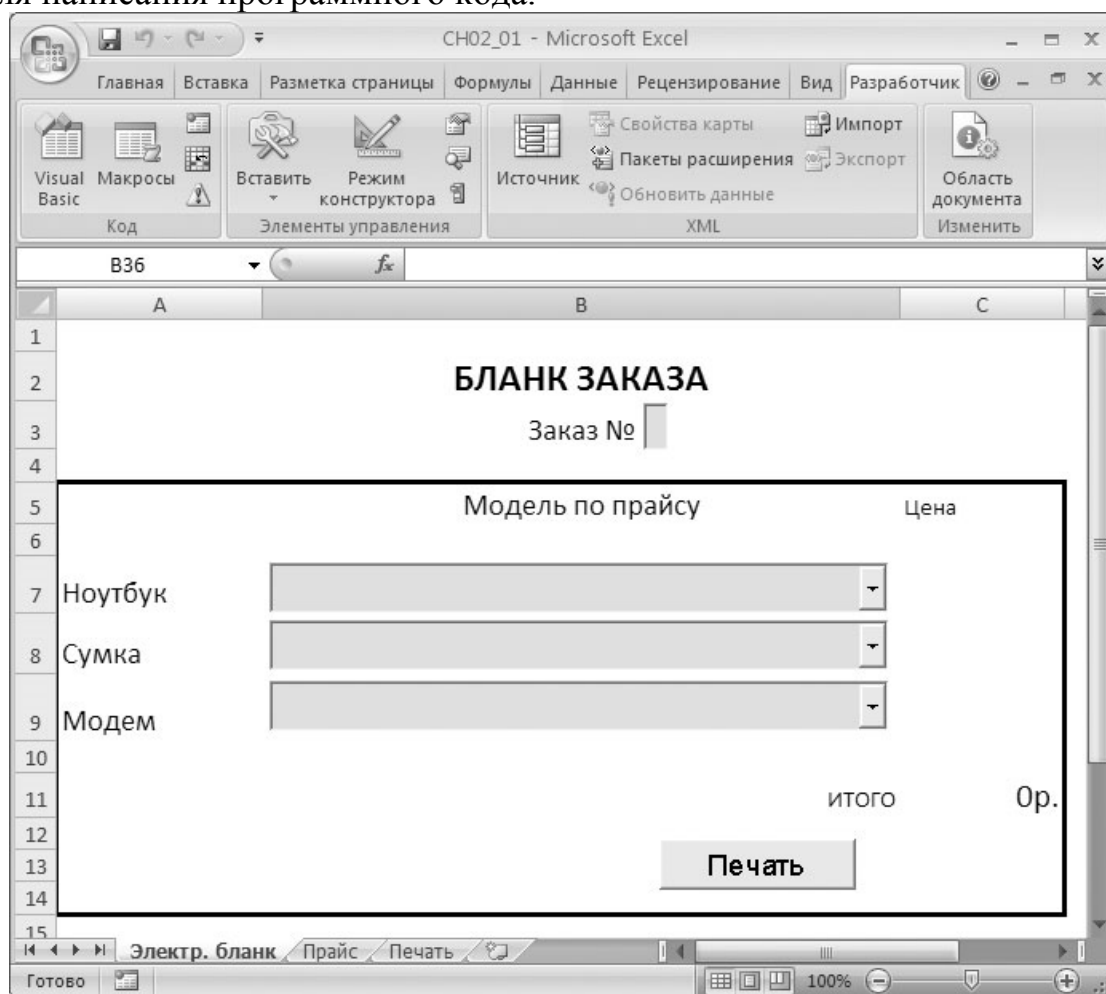


Рис. 118. Лист книги с электронным бланком

Первая наша задача заключается в том, чтобы при открытии книги созданные три поля со списком на первом листе автоматически заполнялись информацией из прайс-листа. Из первой части мы уже знаем, что при открытии книги автоматически выполняется предопределенная процедура `Workbook_Open()`. Для того чтобы перейти к написанию этой процедуры, следует в редакторе Visual Basic открыть меню View и выбрать раздел Project Explorer и в появившемся на экране окне дважды щелкнуть на пиктограмме с названием ЭтаКнига. Таким образом, мы получим доступ к методам объекта Workbook.

В соответствии с листингом оформим процедуру `Workbook_Open()`, которая обеспечивает заполнение элементов управления «Поле со списком» моделями ноутбуков, сумок и модемов.

```
' Листинг Процедура, выполняемая при открытии книги
Private Sub Workbook_Open()
' Очистка первого поля со списком
Worksheets(1).Spk1.Clear
' Подсчет в переменной N количества вариантов ноутбуков по
прайсу
N = 0
While Worksheets(2).Cells(N + 2, 1).Value <> ""
    N = N + 1
Wend
' Заполнение первого списка
For i = 2 To N + 2
    Worksheets(1).Spk1.AddItem Worksheets(2).Cells(i, 1).Value
Next
' Очистка второго поля со списком
Worksheets(1).Spk2.Clear
' Подсчет в переменной N количества моделей сумок
N = 0
    While Worksheets(2).Cells(N + 2, 3).Value <> ""
N = N + 1
Wend
' Заполнение второго списка
For i = 2 To N + 2
    Worksheets(1).Spk2.AddItem Worksheets(2).Cells(i, 3).Value
Next
' Очистка третьего списка на первом листе нашей книги
Worksheets(1).Spk3.Clear
' Подсчет в переменной N количества моделей модемов
N = 0
While Worksheets(2).Cells(N + 2, 5).Value <> ""
    N = N + 1
Wend
' Заполнение третьего списка
```

```

For i = 2 To N + 2
    Worksheets(1).Spk3.AddItem Worksheets(2).Cells(i, 5).Value
Next
'Установка начальных параметров
Worksheets(1).Spk1.ListIndex = -1
Worksheets(1).Spk2.ListIndex = -1
Worksheets(1).Spk3.ListIndex = -1
Worksheets(1).Nomer.Text = ""
Worksheets(1).Range("C7:C9").Value = ""
End Sub

```

Прокомментируем еще несколько программных конструкций этой процедуры.

Так, в записи **While Worksheets (2).Cells (N + 2, 1) .Value <> ""** извлекается содержимое ячейки первого столбца с информацией об очередной модели ноутбука со второго листа. Конструкция **<>** обозначает «не равно», а **""** — это две пары кавычек, между которыми ничего нет. В совокупности эта запись обозначает выполнение цикла, пока значение в очередной ячейке списка ноутбуков на втором листе не окажется пустым. Пустая ячейка — это индикатор того, что данных в этом столбце больше нет. Таким образом, чтобы избежать при заполнении списков потери информации, следует не допускать разрывов в данных на втором листе.

В процедуре листинга после подсчета количества ноутбуков используется метод **AddItem**, который добавляет в элемент управления «Поле со списком» новый элемент (новую строку). Так, в конструкции **Worksheets(1).Spk1.AddItem Worksheets(2).Cells(i, 1).Value** через пробел записывается содержимое ячейки, которое включается в список **Spk1** в качестве его очередного элемента. В данном случае это содержимое ячейки, расположенной на втором листе. Заполнение других списков производится аналогичным образом и различается только номерами столбцов. Таким образом, мы создали процедуру **Workbook\_Open()**, которая обеспечивает заполнение списков данными со второго листа (из прайс-листа) при открытии книги. Теперь, для того чтобы все то, что мы создали, заработало, необходимо книгу сохранить, закрыть и заново ее открыть.

Видно, что списки заполнены, однако пока выбор в списках той или иной модели не приводит к заполнению информации о ее цене. Нашу разработку следует продолжить, поэтому мы заново вернемся в окно написания программного кода. Для автоматической подстановки цены модели необходимо написать процедуры, выполняемые при щелчках на элементах «Поле со списком». В листинге приведена процедура, которая выполняется при щелчке на первом поле со списком (это список, включающий модели ноутбуков).

```

1      ' Листинг  Обработка щелчка на поле
2      ' со списком, включающем модели ноутбуков
3      Private Sub Spk1_Click()
4      Range("C7").Value = _

```

```

5      Worksheets(2).Cells(Spk1.ListIndex + 2, 2).Value
6      End Sub

```

В этой процедуре ячейка C7 заполняется ценой ноутбука из прайс-листа. Если посмотреть на бланк заказа, то именно ячейка C7 на этом листе отводится для цены ноутбука. Здесь Spk1.ListIndex — номер выделенного элемента в первом списке. Двойка добавляется к этому значению потому, что индексация элементов в поле со списком начинается с нуля, а данные о моделях ноутбуков на втором листе располагаются со второй строки. Здесь также учитывается, что цены ноутбуков указаны во втором столбце прайс-листа. Таким образом, в рассмотренной строке мы извлекаем информацию о цене ноутбука. Может возникнуть вопрос — почему в правой части программной строки мы указали номер листа (2), а в левой нет? Это связано с тем, что объект с именем Spk1, событие, связанное с которым мы обрабатываем, находится на первом листе, который рассматривается процедурой в качестве рабочего листа по умолчанию. Щелчки на двух других списках приводят к аналогичным действиям (к заполнению цен выбранных сумки и модема), и в листинге приведены процедуры, реализующие этот эффект.

```

1      ' Листинг. Процедуры выбора цен сумки и модема
2      Private Sub Spk2_Click()
3          Range("C8").Value = _
4              Worksheets(2).Cells(Spk2.ListIndex + 2, 4).Value
5      End Sub
6
7      Private Sub Spk3_Click()
8          Range("C9").Value = _
9              Worksheets(2).Cells(Spk3.ListIndex + 2, 6).Value
10     End Sub

```

В результате мы получили автоматизированное заполнение бланка заказа (выбор в списках названий позиций товаров и автоматическую подстановку цены). Для калькуляции суммы заказа осталось только в ячейку C11 поставить формулу для вычисления суммы: =СУММ(C7:C9). Заметим, что это обыкновенная формула в ячейке первого листа Microsoft Excel.

На рис. 119 показано заполнение бланка заказа выборкой товаров. Следующий этап автоматизации заключается в создании печатной формы этого бланка. Сначала разместим на первом листе кнопку Печать (см. рис. 119), а затем для нее напишем необходимую процедуру. Но в первую очередь мы должны на третьем листе книги создать печатную форму, которая будет автоматически заполняться по щелчку на кнопке. Первая задача на этом пути, как и прежде, чисто техническая — обеспечить форматирование третьего листа книги подобно тому, как это показано на рис. 120 (никаких элементов управления здесь нет). Форматирование листа: добавление фона к ячейкам, создание внешних и внутренних границ, подбор размера шрифта,



выравнивание содержимого ячейки по горизонтали. Также здесь мы убрали отображение сетки. В дальнейшем в ячейку C2 будет подставляться номер заказа с первого листа (точнее, из соответствующего текстового окна), а содержимое таблицы также будет автоматически заполняться информацией о заказанных позициях и их ценах.

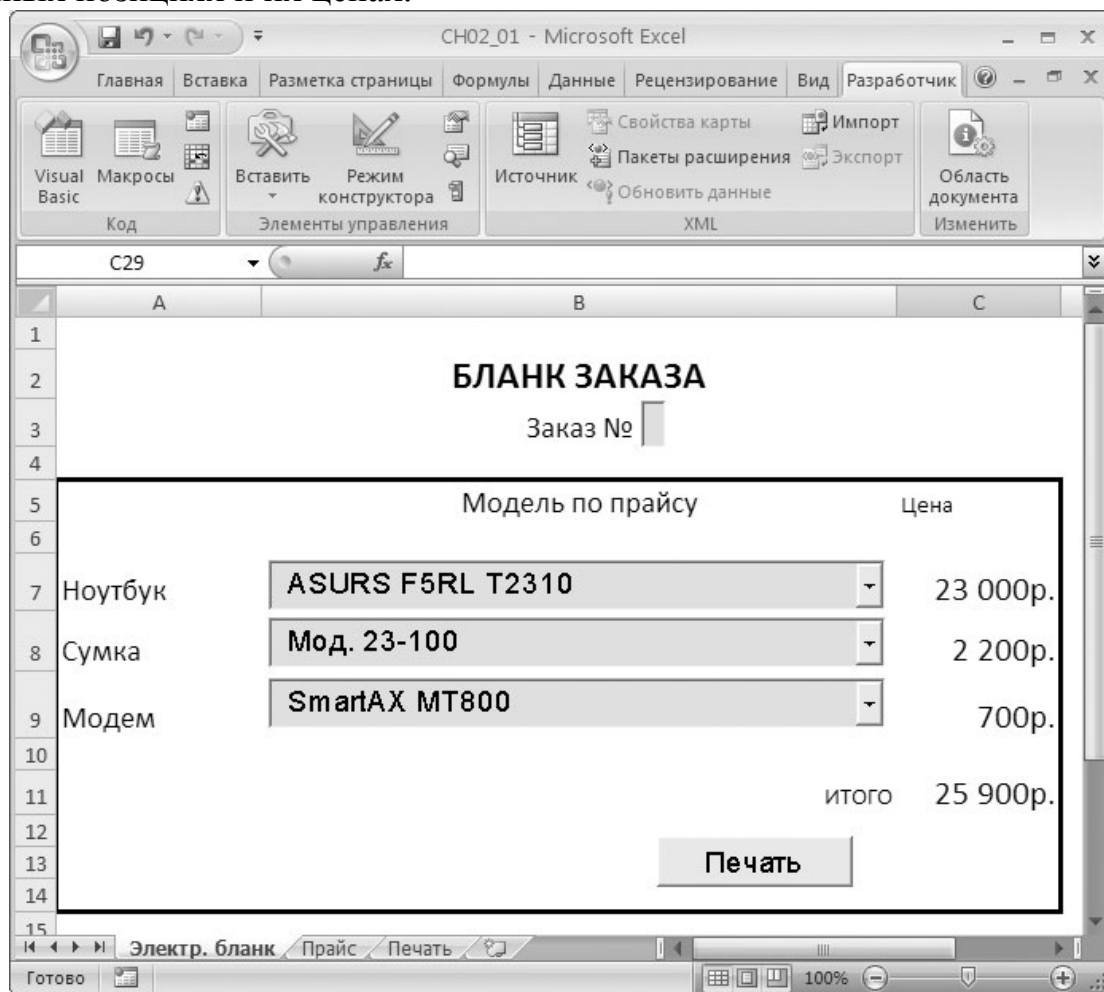


Рис. 119. Вариант заполнения электронного бланка

При желании на печатную форму можно добавить логотип организации и внести дополнительную текстовую информацию. В этом случае третий лист нашей книги станет больше похож на печатную форму документа. Фактически мы разработали заготовку, в которую по щелчку на кнопке Печать будет заноситься информация о текущем заказе с первого листа. Перейдем в режим конструктора и далее в редактор написания программного кода. Это выполняется, как мы знаем, двойным щелчком на кнопке Печать. В этом случае вы автоматически попадаете в процедуру, которая обрабатывает щелчок на рассматриваемой кнопке. В листинге 2.4 приводится текст этой процедуры.

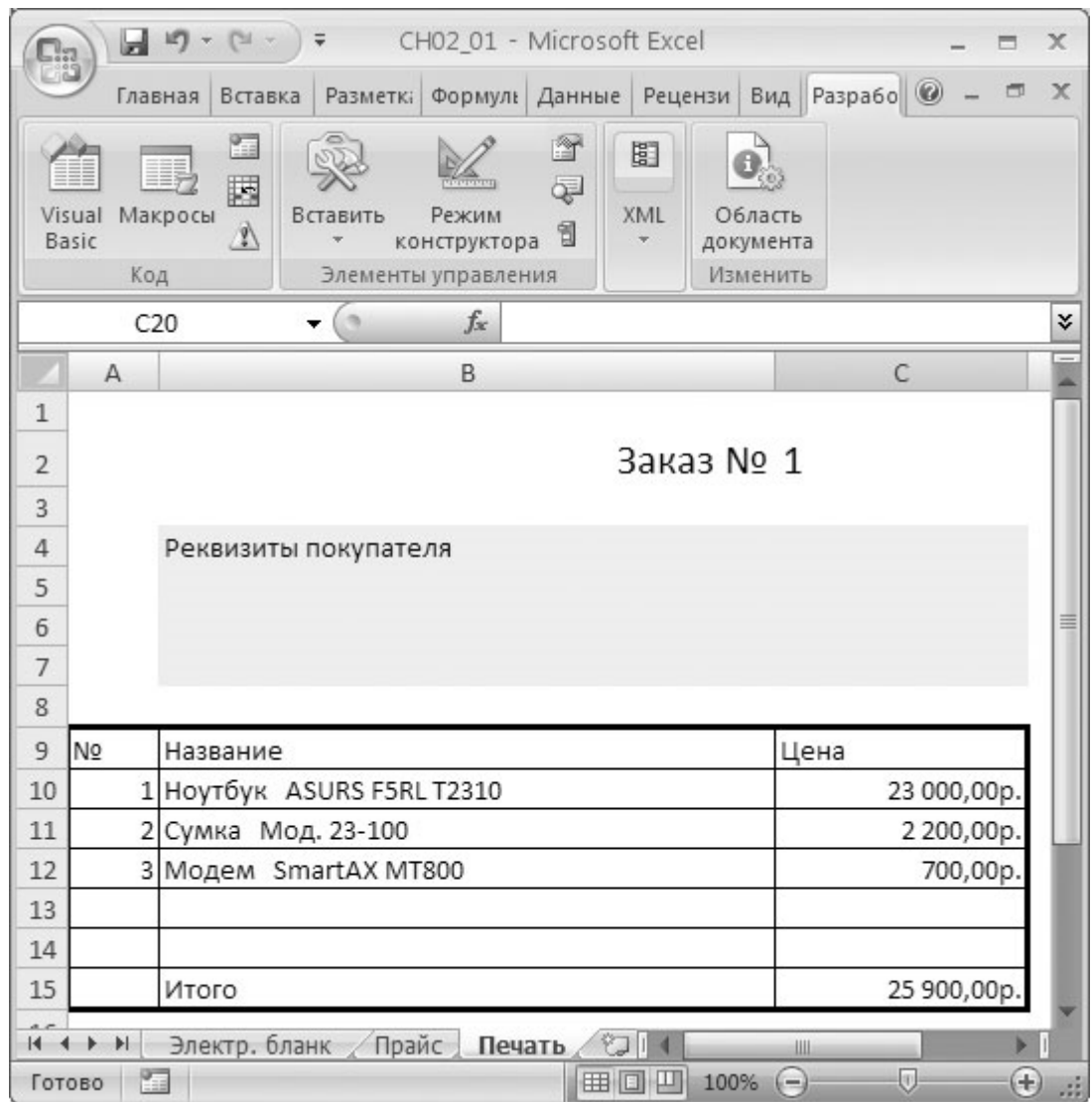


Рис. 120. Печатная форма бланка

```
Private Sub Prn_Click()
```

```
' Очистка табличной части печатного бланка
```

```
Worksheets(3).Range("A10:C15").Value = ""
```

```
Nom = 1
```

```
' Внесение в печатную форму информации о ноутбуке
```

```
If Spk1.Text <> "" Then
```

```
Worksheets(3).Cells(9 + Nom, 1).Value = Nom
```

```
Worksheets(3).Cells(9 + Nom, 2).Value = _
```

```
Worksheets(1).Cells(7, 1).Value + "" + Spk1.Text
```

```
Worksheets(3).Cells(9 + Nom, 3).Value = _
```

```
Worksheets(2).Cells(Spk1.ListIndex + 2, 2).Value
```

```
Nom = Nom + 1
```

```
End If
```

```
' Внесение в печатную форму информации о сумке к ноутбуку
```

```
If Spk2.Text <> "" Then
```

```
Worksheets(3).Cells(Nom + 9, 1).Value = Nom
```

```
Worksheets(3).Cells(Nom + 9, 2).Value = _
```

```
Worksheets(1).Cells(8, 1).Value + "" + Spk2.Text
```

```

Worksheets(3).Cells(Nom + 9, 3).Value = _
Worksheets(2).Cells(Spk2.ListIndex + 2, 4).Value
Nom = Nom + 1
End If
' Внесение в печатную форму информации о модеме
If Spk3.Text <> "" Then
Worksheets(3).Cells(Nom + 9, 1).Value = Nom
Worksheets(3).Cells(Nom + 9, 2).Value = _
Worksheets(1).Cells(9, 1).Value + "" + Spk3.Text
Worksheets(3).Cells(Nom + 9, 3).Value = _
Worksheets(2).Cells(Spk3.ListIndex + 2, 6).Value
Nom = Nom + 1
End If
' Установка номера заказа
Worksheets(3).Range("C2").Value = Nomer.Text
Worksheets(3).Range("B15").Value = "uToro"
' Перенос итоговой суммы
Worksheets(3).Range("C15").Value = Range("C11").Value
Worksheets(3).Activate
End Sub

```

Теперь автоматизированная книга готова, и вы можете попробовать оформить с помощью созданной разработки несколько гипотетических заказов.

## Лабораторная работа №11 Модернизация бланка заказа

В предыдущей работе мы рассмотрели один из вариантов формирования бланка заказа. Понятно, что в многочисленных практических ситуациях требуются различные бланки. Рассмотрим разработку автоматизированного бланка вида, показанного на рис. 121. В этом случае прайс-лист формируется из товаров общей категории (рис. 122), и при открытии книги все поля со списками на первом листе книги заполняются одинаковым набором товаров. В отличие от предыдущей работы, в книгу добавлена дополнительная функциональность — столбец для указания количества единиц выбранного товара. В другом столбце (Е) подсчитывается сумма по каждому товару, указанному в бланке — это обычная формула в ячейке (произведение стоимости единицы товара на количество). Такая организация бланка позволяет получить в заказе несколько однотипных товаров (например, несколько сумок).

The screenshot shows a Microsoft Excel window titled 'CH02\_05 - Microsoft Excel'. The 'Разработчик' (Developer) tab is active. The worksheet contains a form titled 'БЛАНК ЗАКАЗА' (Order Form). At the top, there is a text box for 'Заказ №' (Order No.) with the value '1'. Below this is a table with the following columns: 'Номер' (Number), 'Модель по прайсу' (Model by price), 'Цена' (Price), 'Количество' (Quantity), and 'Сумма' (Sum). The table contains three rows of data: 1. 'ACER 4568' with price '23 000р.' and quantity '1', sum '23 000р.'. 2. 'ACER 4569' with price '27 000р.' and quantity '2', sum '54 000р.'. 3. 'Сумка (мод. 23-100)' with price '2 200р.' and quantity '2', sum '4 400р.'. Below the table, there is a row for 'Итого' (Total) with a sum of '81 400р.'. At the bottom right of the table area, there is a 'Печать' (Print) button. The bottom status bar shows 'Готово' (Ready) and '100%' zoom.

Номер	Модель по прайсу	Цена	Количество	Сумма
1	ACER 4568	23 000р.	1	23 000р.
2	ACER 4569	27 000р.	2	54 000р.
3	Сумка (мод. 23-100)	2 200р.	2	4 400р.
Итого				81 400р.

Рис. 121. Модернизированный вариант бланка

В листинге представлена процедура, выполняемая при открытии книги. С учетом предыдущих разработок, строки программы не требуют дополнительного комментария. Заметим только, что имена списков выбраны следующим образом — Spk1, Spk2, Spk3, Spk4, Spk5. В отличие от только что рассмотренного бланка, в этом случае списки заполняются одинаково — всеми позициями номенклатуры, которые имеются в прайс-листе.

```
' Листинг Процедура, выполняемая при открытии книги  
Private Sub Workbook_open()
```

```

' Очистка списков на первом листе книги
Worksheet(1).Spk1.Clear
Worksheet(1).Spk2.Clear
Worksheet(1).Spk3.Clear
Worksheet(1).Spk4.Clear
Worksheet(1).Spk5.Clear
' Подсчет в переменной N количества товаров по прайсу
N = 0
While Worksheets(2).Cells(N + 2, 1).Value <> ""
    N = N + 1
Wend
' Заполнение списков
For i = 2 To N + 1
    Worksheets(1).Spk1.AddItem Worksheets(2).Cells(i, 1).Value
    Worksheets(1).Spk2.AddItem Worksheets(2).Cells(i, 1).Value
    Worksheets(1).Spk3.AddItem Worksheets(2).Cells(i, 1).Value
    Worksheets(1).Spk4.AddItem Worksheets(2).Cells(i, 1).Value
    Worksheets(1).Spk5.AddItem Worksheets(2).Cells(i, 1).Value
Next
' Установка начальных параметров
Worksheets(1).Spk1.ListIndex = -1
Worksheets(1).Spk2.ListIndex = -1
Worksheets(1).Spk3.ListIndex = -1
Worksheets(1).Spk4.ListIndex = -1
Worksheets(1).Spk5.ListIndex = -1
Worksheets(1).Nomer.Text = ""
Worksheets(1).Range("A7:A11").Value = ""
Worksheets(1).Range("C7:D11").Value = ""
End Sub

```

После открытия книги ее функциональность обеспечивается процедурами, выполняемыми по щелчкам на списках товаров — когда мы выбираем конкретный товар.

В листинге представлена процедура, выполняемая при щелчке на первом списке (самом верхнем). Здесь первая строка обеспечивает выбор цены из прайс-листа для соответствующего товара. Следующие строки — установку единицы в качестве номера строки и единицы в качестве количества единиц выбранной номенклатуры. Для автоматического вычисления суммы в ячейке E7 установим формулу **=C7\*D7**.

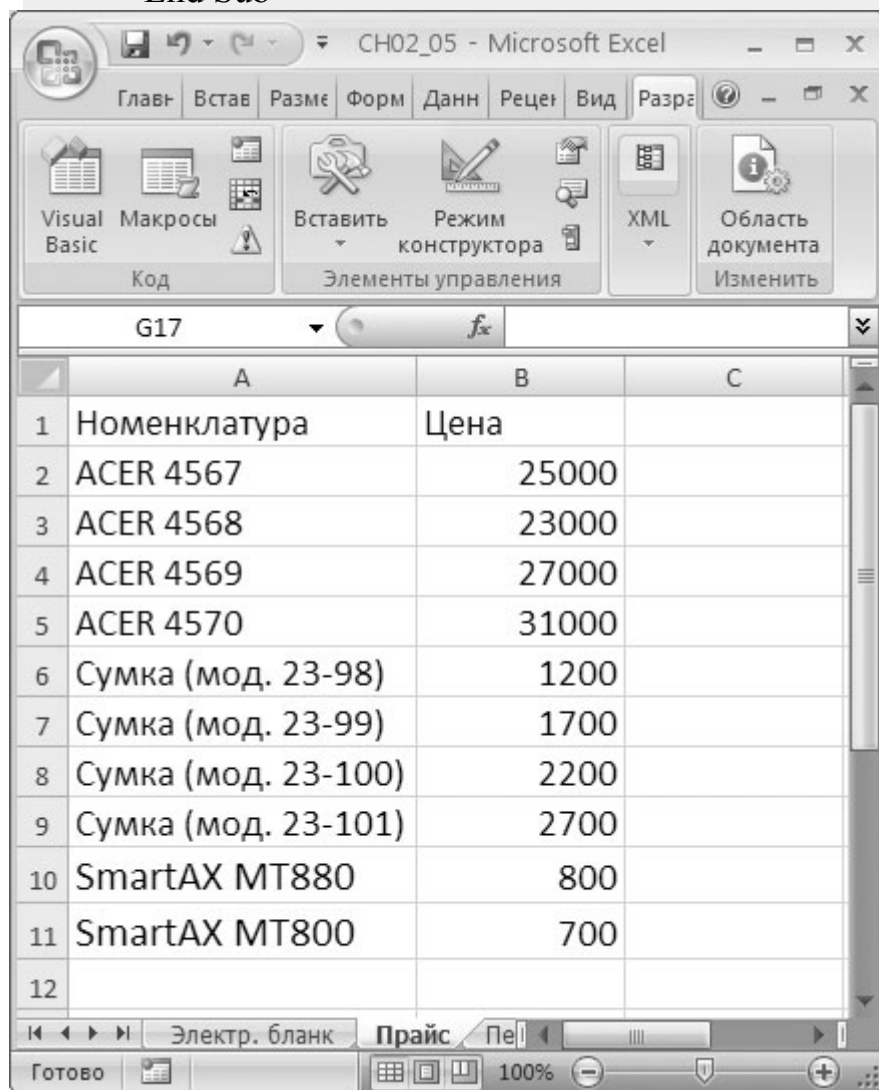
В реальной работе часто требуется отменять принятое решение — отменять выбранную позицию товара. Для реализации этого мы использовали процедуру **Spk1\_Change**, которая автоматически выполняется при изменениях в поле со списком. На рис. 122 приведен текст данной процедуры, который обеспечивает при удалении информации из поля со списком сброс значений в ячейках A7, C7, D7.

```

1      ' Листинг Процедура, выполняемая по щелчку на первом
2списке
3      Private Sub Spk1_Click()
4      Range("C7").Value = _
5      Worksheets(2).Cells(Spk1.ListIndex + 2, 2).Value
6      Range("A7").Value = 1
7      Range("D7").Value = 1
      End Sub

1      ' Листинг Процедура, выполняемая при изменениях в первом
2списке
3      Private Sub Spk1_Change()
4      If Spk1.Text = "" Then
5          Range("A7").Value = ""
6          Range("C7").Value = ""
7          Range("D7").Value = ""
8      End If
      End Sub

```



	A	B	C
1	Номенклатура	Цена	
2	ACER 4567	25000	
3	ACER 4568	23000	
4	ACER 4569	27000	
5	ACER 4570	31000	
6	Сумка (мод. 23-98)	1200	
7	Сумка (мод. 23-99)	1700	
8	Сумка (мод. 23-100)	2200	
9	Сумка (мод. 23-101)	2700	
10	SmartAX MT880	800	
11	SmartAX MT800	700	
12			

Рис. 122. Общий прайс-лист товаров

Таким образом, процедуры, приведенные в листингах, обеспечивают работу пользователя с первой строкой бланка заказа. В листингах приведены аналогичные процедуры для второй строки бланка.

```

1      ' Листинг. Процедура, выполняемая по щелчку на втором
2списке
3      Private Sub Spk2_Click()
4      Range("C8").Value = _
5      Worksheets(2).Cells(Spk2.ListIndex + 2, 2).Value
6      Range("A8").Value = 2
7      Range("D8").Value = 1
8      End Sub

1      ' Листинг Процедура, выполняемая при изменениях во втором
2списке
3      Private Sub Spk2_Change()
4      If Spk2.Text = "" Then
5          Range("A8").Value = ""
6          Range("C8").Value = ""
7          Range("D8").Value = ""
8      End If
9      End Sub

```

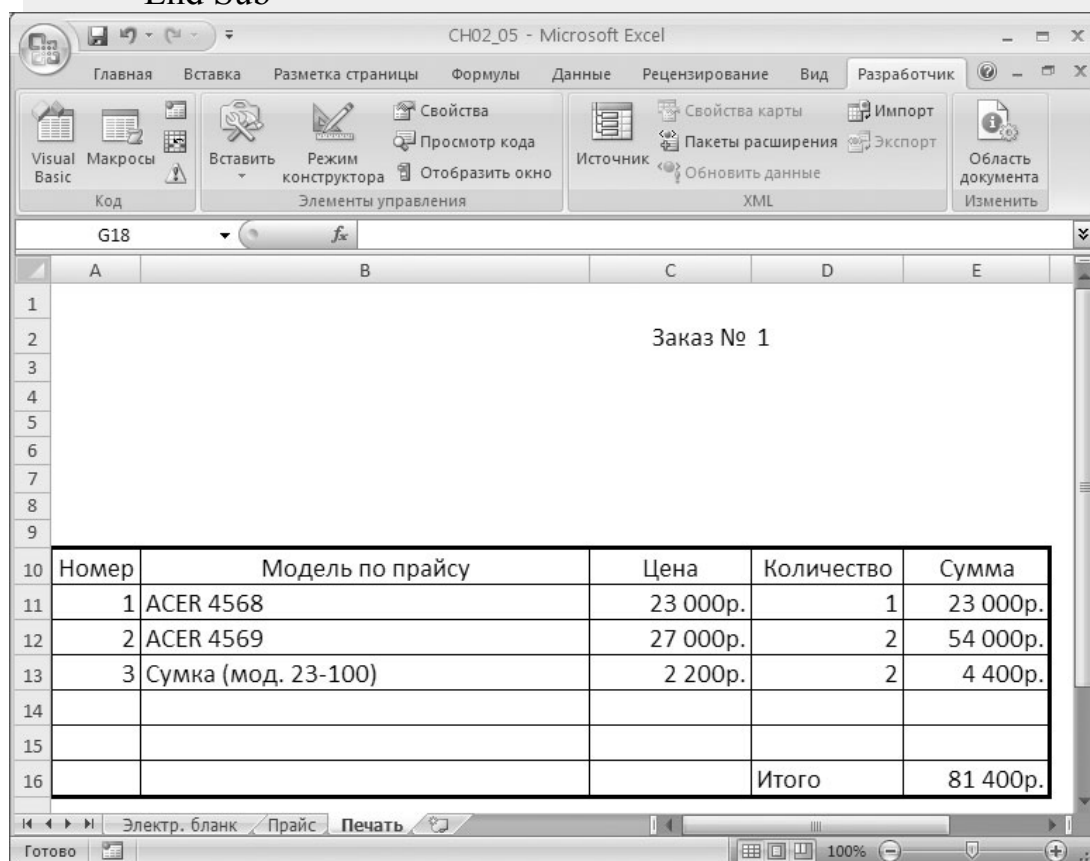


Рис. 123. Печать бланка заказа

Осталось обеспечить печать бланка (рис. 123). В листинге приведена процедура, которая обеспечивает заполнение печатной формы.

```

' Листинг Процедура печати бланка
Private Sub Prn_Click()

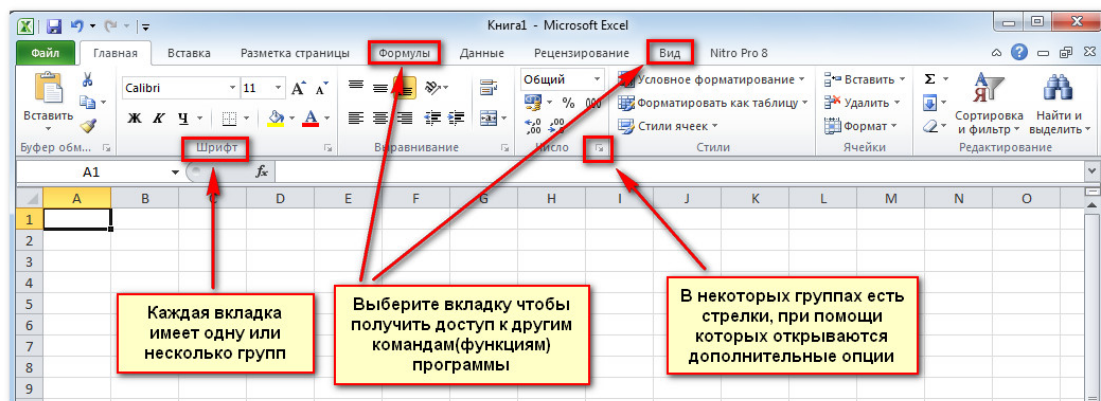
```

```
' Очистка табличной части печатной формы
Worksheets(3).Range("A11:E16").Value = ""
' Перенос информации с бланка в печатную форму
For i = 1 To 5
    Worksheets(3).Cells(i + 10, 1).Value = Cells(6 + i, 1).Value
    Worksheets(3).Cells(i + 10, 3).Value = Cells(6 + i, 3).Value
    Worksheets(3).Cells(i + 10, 4).Value = Cells(6 + i, 4).Value
    Worksheets(3).Cells(i + 10, 5).Value = Cells(6 + i, 5).Value
Next
Worksheets(3).Cells(11, 2).Value = Spk1.Text
Worksheets(3).Cells(12, 2).Value = Spk2.Text
Worksheets(3).Cells(13, 2).Value = Spk3.Text
Worksheets(3).Cells(14, 2).Value = Spk4.Text
Worksheets(3).Cells(15, 2).Value = Spk5.Text
Worksheets(3).Range("D2").Value = Nomer.Text
Worksheets(3).Range("D16").Value = "ИТОГО"
Worksheets(3).Range("E16").Value = Range("E12").Value
Worksheets(3).Activate
End Sub
```



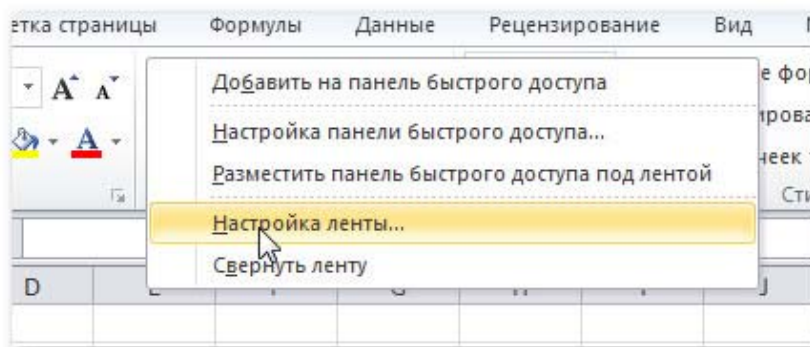
## Лабораторная работа №12 Создание собственного головного меню

Лента содержит несколько вкладок, на каждой из которых несколько групп команд. Вы можете добавлять свои собственные вкладки с вашими любимыми командами.

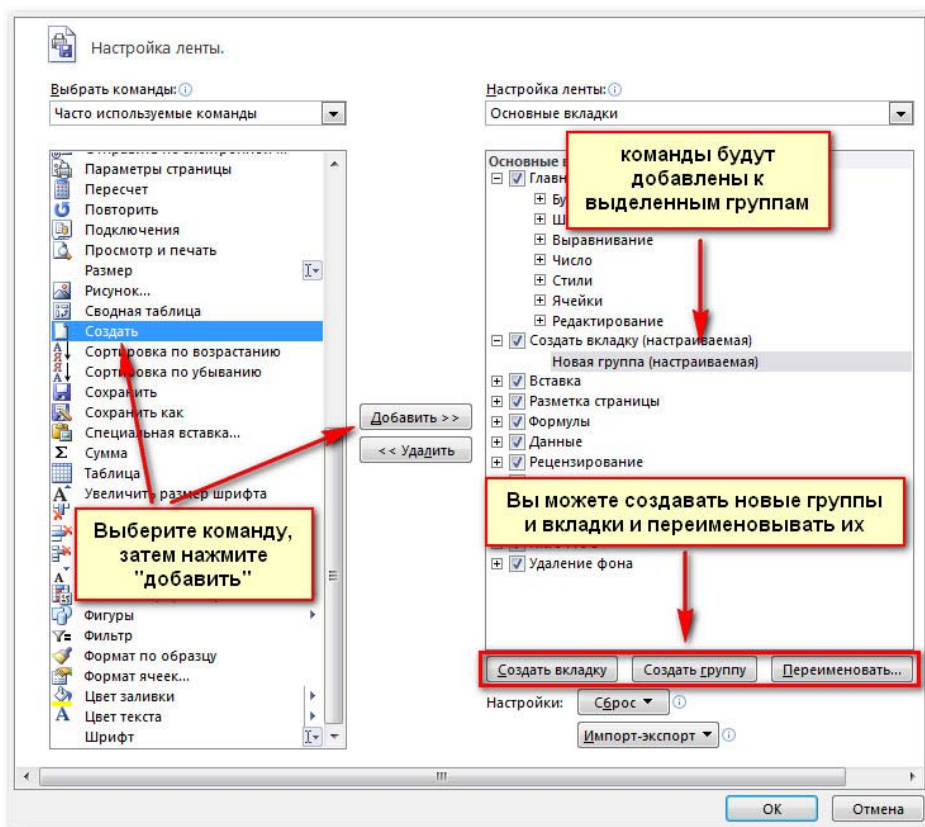


Чтобы настроить Ленту: вы можете настроить Ленту, создав свои собственные вкладки с нужными командами. Команды всегда располагаются в группе. Вы можете создать так много групп, как вам нужно. Более того, вы можете добавлять команды на вкладки, которые имеются по умолчанию, при условии, что вы создадите для них группу.

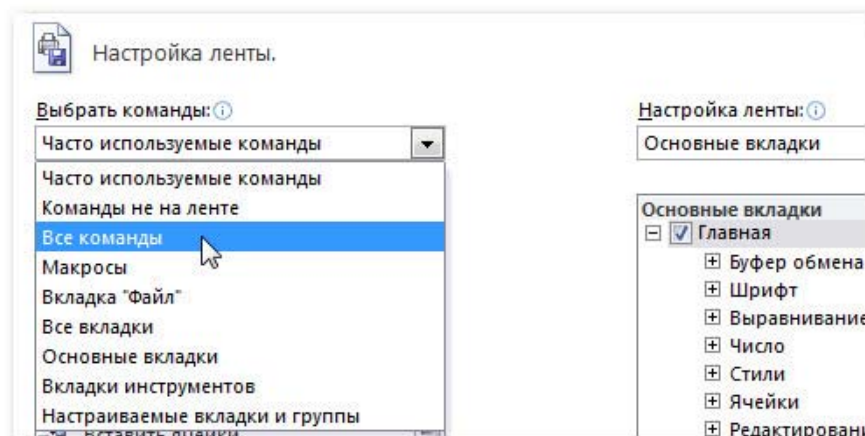
1. Кликните по Ленте правой кнопкой мыши и выберите Настройка ленты. Откроется диалоговое окно.



2. Нажмите Новая вкладка. Будет создана новая вкладка с новой группой внутри.
3. Убедитесь, что выбрана новая группа.
4. В списке слева выберите команду и нажмите Добавить. Вы также можете просто перетащить команду в группу.
5. Когда вы добавите все нужные команды, нажмите ОК.



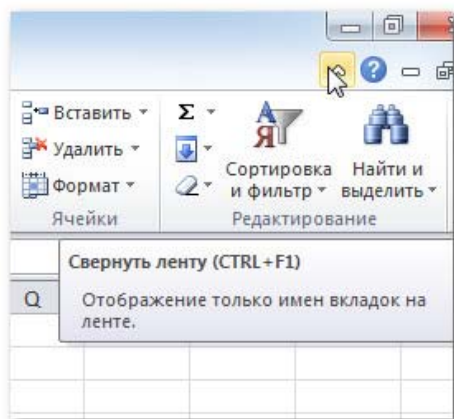
Если вы не можете найти нужную команду, кликните по выпадающему списку **Выбрать команды** и выберите **Все команды**.



Чтобы свернуть и развернуть Ленту:

Лента призвана оперативно реагировать на ваши текущие задачи и быть легкой в использовании. Тем не менее, вы можете ее свернуть, если она занимает слишком много экранного пространства.

1. Кликните по стрелке в правом верхнем углу Ленты, чтобы ее свернуть.
2. Чтобы развернуть Ленту кликните по стрелке снова.



Когда лента свернута, вы можете временно ее отобразить, нажав на какую-либо вкладку. Вместе с тем, лента снова исчезнет, когда вы прекратите ее использовать.

### Панель быстрого доступа

Панель быстрого доступа расположена над Лентой и дает доступ к некоторым нужным командам вне зависимости от того, на какой вкладке вы сейчас находитесь. По умолчанию показываются команды Сохранить, Отменить, Вернуть. Вы можете добавить команды, чтобы сделать панель более удобной для себя.

Чтобы добавить команды на Панель быстрого доступа:

1. Кликните по стрелке справа на панели быстрого доступа.
2. В появившемся списке выберите команды, которые нужно добавить. Чтобы выбрать отсутствующие в списке команды, нажмите Другие команды.

