

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ

**ПРОГРАММИРОВАНИЕ НА VISUAL BASIC
FOR APPLICATIONS В EXCEL**

Учебное пособие

КАЗАНЬ
2012

УДК 519.682

*Печатается по решению
заседания учебно-методической комиссии
Института Вычислительной математики и информационных технологий
Протокол №11 от 7 июня 2012 года*

*заседания кафедры информатики и вычислительных технологий
протокол №10 от 17 мая 2012 г.*

*Авторы-составители:
кандидат физ. мат. наук, доц. О.А. Широкова
ст. преп. Р.Ш. Гайнанова*

*Научный редактор
кандидат физ. мат. наук, доцент Л.У. Бахтиева (КФУ)*

*Рецензенты:
кандидат физ. мат. наук, доц. Н.А. Иваньшин (КГАСА)
кандидат физ. мат. наук, доц. Е.Г. Чепкунова (КФУ)*

Программирование на Visual Basic for Applications в Excel: учебное пособие /
Р. Ш. Гайнанова, О. А. Широкова – Казань: КФУ, 2012. – 153с.

Предназначено для аудиторных занятий и самостоятельных работ студентов, обучающихся по специальностям «Информатика» и «Прикладная информатика». Каждый раздел содержит теоретический материал, примеры решения задач с комментариями и задания для самостоятельной работы.

Может быть использовано при изучении курса «Компьютерное моделирование» и различных курсов по выбору.

© Казанский (Приволжский)
федеральный университет, 2012

Содержание

Введение	4
1. Основные элементы VBA.....	4
1.1. Структура редактора VBA	4
1.2. Элементы управления	8
2. Типы данных, операции, встроенные функции.....	17
3. Управляющие конструкции VBA.....	27
3.1. Оператор присваивания	27
3.2. Оператор безусловного перехода.....	29
3.3. Операторы ветвления	29
3.4. Операторы цикла	40
4. Строки	47
5. Массивы.....	52
5.1. Одномерные массивы.....	52
5.2. Двумерные массивы	53
5.3. Динамические массивы	61
6. Процедуры и функции	65
6.1. Описание и вызов процедур и функций.....	65
6.2. Создание пользовательских функций	68
6.3. Создание процедур обработки ошибочных ситуаций	77
7. Основные объекты VBA в Excel.....	86
7.1. Объект Application.....	86
7.2. Объект Workbook и семейство Workbooks.....	87
7.3. Объект Worksheet и семейство Worksheets	88
7.4. Объекты Range и Selection.....	90
7.5. Объект ActiveCell	93
8. Методы объекта Range, использующие команды Excel.....	100
8.1. Метод GoalSeek.....	100
8.2. Метод AutoFill.....	105
8.3. Метод Sort	107
8.4. Метод AutoFilter	109
9. Создание макросов.....	118
10. Компьютерное моделирование физических задач средствами VBA в Excel	126
10.1. Задача о свободном падении тела с учетом сопротивления среды	127
10.2. Задача о движении тела, брошенного под углом к горизонту, с учетом сопротивления среды.....	138
Литература.....	153

Введение

Visual Basic for Applications – это объектно-ориентированный язык программирования, который используется в качестве макроязыка в электронных таблицах, текстовых процессорах и других приложениях. Отличительной особенностью VBA является использование наряду с обычными переменными и константами также и имеющихся объектов приложений Microsoft Office, например, в Microsoft Office Excel это могут быть рабочие книги, рабочие листы, диапазоны ячеек, диаграммы и т.д. С помощью VBA можно разрабатывать программы, которые включают компоненты нескольких приложений Microsoft Office и способствуют тем самым интеграции и совместному использованию данных.

VBA использует технологию визуального программирования, т.е. конструирования рабочей поверхности приложения и элементов его управления непосредственно на экране.

1. Основные элементы VBA


1.1. Структура редактора VBA

Код VBA создается в редакторе Visual Basic Editor. Чтобы запустить этот редактор, необходимо перейти на вкладку Разработчик (рис. 1.1) и в группе Код нажать кнопку Visual Basic: откроется интегрированная среда разработки приложений Visual Basic (рис. 1.3).

Если на ленте в Microsoft Office Excel 2010 нет вкладки **Разработчик**, необходимо выполнить следующие действия:

1. Перейти на вкладку **Файл** и нажать на кнопку **Параметры**.
2. В открывшемся окне **Параметры Excel** выбрать слева категорию **Настройка ленты**, а справа в группе **Настройка ленты** - из раскрывающегося списка **Основные вкладки**.
3. Установить флажок **Разработчик** (рис. 1.2) и нажать на кнопку **ОК**.

Примечание

Для быстрого запуска редактора VBA достаточно нажать комбинацию клавиш <Alt>+<F11>. Возвратится из редактора VBA в рабочую книгу можно нажатием кнопки **View Microsoft Excel** .

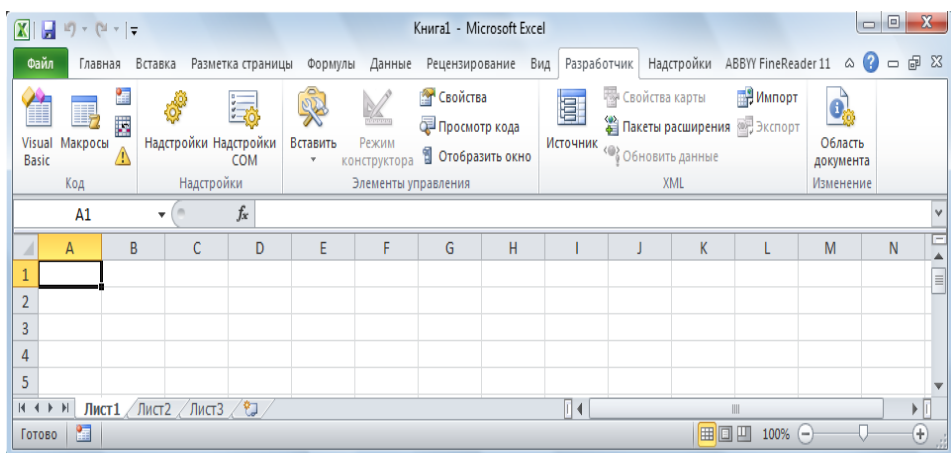


Рис. 1.1. Вкладка **Разработчик** на ленте

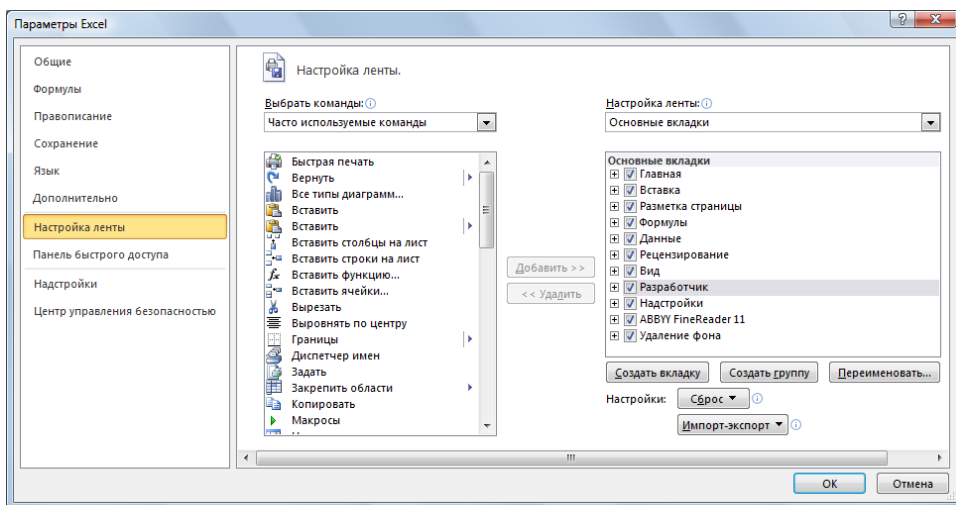




Рис. 1.2. Окно **Параметры Excel**

Интерфейс VBA состоит из следующих основных компонентов: окно проекта, окно свойств, окно редактирования кода, окно форм, меню и панели инструментов.

Окно проекта в редакторе VBA активизируется выбором команды

View, Project Explorer или нажатием кнопки **Project Explorer** . В окне проекта (VBAProject) перечислены все входящие в проект

объекты. По умолчанию в проект VBA входят три рабочих листа и рабочая книга.

В окне свойств отображаются все свойства выбранного объекта. Используя это окно можно просматривать свойства или изменять их установки. Для просмотра свойств выбранного объекта надо либо щелкнуть на кнопку **Properties Window** , либо выбрать команду **View, Properties Window**. Окно свойств состоит из двух составных частей: верхней и рабочей. В верхней части окна свойств располагается раскрывающийся список, из которого можно выбрать любой, входящий в проект объект. Рабочая часть состоит из двух вкладок: **Alphabetic** (по алфавиту) и **Categorized** (по категориям), отображающие набор свойств по алфавиту или по категориям.

В проект можно добавлять дополнительные объекты. Код программы, который не связан с определенным объектом (например, с листом, формой или объектом формы), обычно хранится в *модуле* (module). Модуль – это объект, в котором хранится код программы с определением переменных, подпрограмм и функций. Для добавления модуля в проект нужно выполнить команду **Insert, Module**. После того как в проект добавлен модуль, в области разработки отобразится код, хранящийся в этом модуле (рис. 1.3).

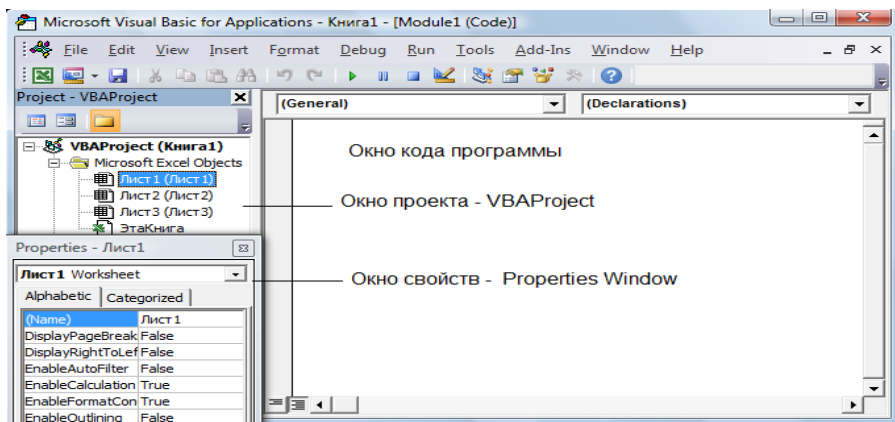


Рис. 1.3.

При написании кода редактор предлагает пользователю список компонентов, логически завершающих вводимую пользователем инструкцию. Например, при наборе кода

TextBox1.T

после ввода буквы Т отобразится список компонентов (рис. 1.4), которые логически завершают данную инструкцию. Двойной щелчок на выбранном элементе из этого списка или нажатие клавиши <Tab> вставляет выбранное имя в код программы.

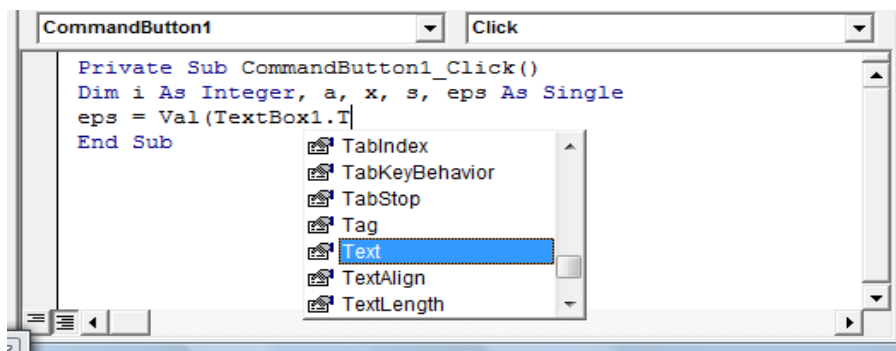



Рис. 1.4. Список компонентов

Для создания диалоговых окон в приложениях VBA используются формы. Редактор форм является одним из основных инструментов визуального программирования. Форма в проект добавляется с помощью команды **Insert, UserForm** или нажатием кнопки **Insert UserForm** . В результате на экран выводится незаполненная форма с панелью элементов (Toolbox) (рис. 1.5). Используя панель элементов из незаполненной формы можно сконструировать любое требуемое для приложения диалоговое окно.

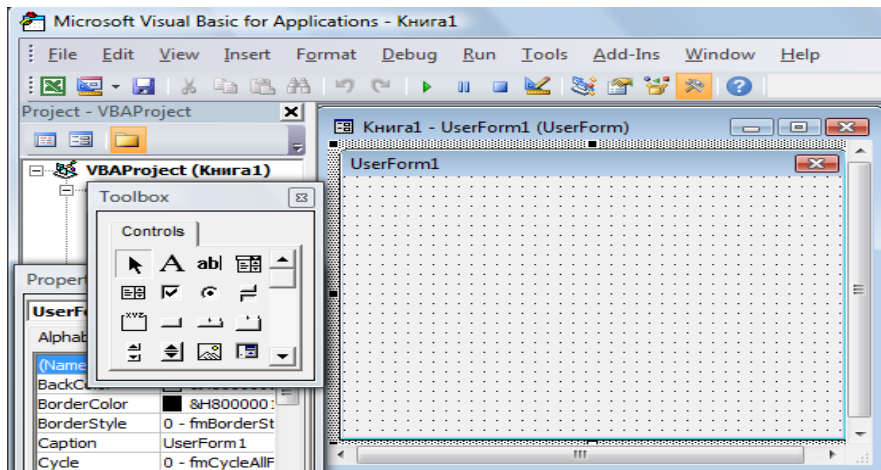


Рис. 1.5. Окно редактирования форм и панель элементов

1.2. Элементы управления

VBA обладает встроенным набором элементов управления. Элементы управления являются объектами. Поэтому как любые объекты, они обладают *свойствами, методами, событиями*. Доступные элементы управления отображены на вкладке Controls панели Toolbox (рис. 1.5). Размещение элементов управления на рабочем листе или в форме, как правило, происходит на начальном этапе конструирования приложения. Большинство элементов управления можно располагать как на рабочем листе, так и в форме.

На рабочем листе элементы управления доступны при нажатии кнопки **Вставить**, которая расположена на вкладке **Разработчик** в группе **Элементы управления**. При нажатии кнопки **Вставить** доступны две группы элементов управления: **Элементы управления формы** и **Элементы ActiveX** (рис. 1.6). Группа **Элементы управления формы** предназначена, прежде всего, для обеспечения совместимости с документами ранних версий Excel. **Элементы ActiveX** (ActiveX Controls) являются независимыми компонентами различных приложений и, в том числе, могут использоваться в Excel.

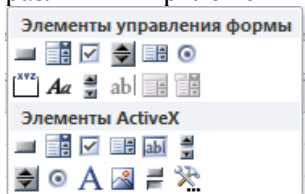



Рис. 1.6.

Кроме стандартных элементов управления можно использовать дополнительные элементы управления. Например, элементы управления мульти-медиа, с помощью которых можно воспроизводить звук или видео с рабочего листа.

Кроме того, существует возможность подключать элементы управления, которые используются в других программах, или отдельно созданные элементы управления. В модуле рабочего листа можно создать процедуры, которые будут обрабатывать те или иные события, происходящие с элементами управления. Например, нажатие кнопки, выбор элемента из списка, выбор переключателя, установка флажка и т.д. будут автоматически приводить к тем или иным вычислениям, построению диаграмм или смене их типа и т.п. Рассмотрим пример использования элемента управления **Кнопка** (CommandButton)  из группы **Элементы ActiveX**, а также события, связанного с ней – **Click**, которое генерируется при нажатии кнопки. Пусть при нажатии на элемент управления **Кнопка**, который мы расположим на рабочем листе **Лист1**, будет выполняться активизация рабочего листа **Лист2**.

	A	B	C	D	E
1					
2					
3					
4					
5					
6					

Рис. 1.7.

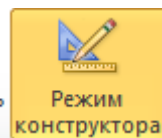



Рис.1.8.

Для этого необходимо:


1. Запустить Microsoft Office Excel 2010 и указатель ячейки установить на рабочем листе **Лист1**.
2. Перейти на вкладку **Разработчик** и в группе **Элементы управления** нажать кнопку со списком **Вставить**.
3. Элемент управления **Кнопка** (CommandButton)  из группы **Элементы ActiveX** установить на рабочем листе. После появления кнопки на рабочем листе на вкладке **Разработчик** кнопка **Режим Конструктора** становится активной (включенной) (рис. 1.8).
4. Щелкнуть на созданной кнопке правой кнопкой мыши и из появившегося контекстного меню выбрать команду **Свойства** для открытия окна свойств (Properties). В окне свойств в поле **Caption** для изменения надписи на кнопке ввести «**Перейти на Лист2**».
5. Создать код процедуры, обрабатывающей событие «нажатие кнопки». В результате обработки этого события должен активизироваться рабочий лист **Лист2**. Для этого: дважды щелкнуть на созданной кнопке (кнопка **Режим Конструктора** в группе **Элементы управления** на вкладке **Разработчик** нажата): откроется редактор VBA с активизированным модулем рабочего листа, в который автоматически добавились первая и последняя инструкции процедуры обработки события «нажатие кнопки» (Click). В этой процедуре набрать инструкцию: `Sheets("Лист2").Select`, в результате получим:

```
Private Sub CommandButton1_Click()
    Sheets("Лист2").Select
End Sub
```
6. Созданная кнопка будет обрабатывать событие только после выхода из режима конструктора. Отключить режим

конструктора нажатием кнопки **Режим Конструктора** в группе **Элементы управления** на вкладке **Разработчик**.


7. Протестировать созданную кнопку (рис. 1.7). Нажатие на эту кнопку должно привести к активизации рабочего листа **Лист2**.

Основные элементы управления

Надпись  **Label** - используется для отображения надписей. Надпись не может быть изменена пользователем, но код программы во время ее выполнения может управлять текстом надписи.

Свойства:


Caption	Возвращает текст, отображаемый в надписи
Visible	Допустимые значения: True (отображается во время выполнения программы) и False (в противном случае)
Multiline	Допустимые значения: True (устанавливается многострочный режим ввода текста в поле) и False (однострочный режим)
WordWrap	Допустимые значения: True (устанавливается режим автоматического переноса) и False (в противном случае)
AutoSize	Допустимые значения: True (устанавливается режим автоматического изменения размера поля так, чтобы весь вводимый текст помещался в нем) и False (устанавливается фиксированный размер поля)
Alignment	выравнивание текста: Left (влево), Right (вправо), Center (по центру)
Font	шрифт, его размер, начертание
ToolTipText	подсказка, появляющаяся при наведении указателя мыши на элемент управления
ForeColor	цвет текста
BackColor	цвет фона


Текстовое поле  **TextBox** - используется для ввода текста, который в последующем используется в программе, или для вывода результатов расчетов программы.

Свойства:

Text	Возвращает текст, содержащийся в поле
ScrollBars	устанавливает режим отображения в поле полос прокрутки. Допустимые значения: <ul style="list-style-type: none">▪ fmScrollBarsNone – не выводить полосы прокрутки▪ fmScrollBarsHorizontal – выводить горизонтальную полосу прокрутки▪ fmScrollBarsVertical – выводить вертикальную полосу прокрутки▪ fmScrollBarsBoth - выводить горизонтальную и вертикальную полосы прокрутки

Другие свойства аналогичны элементу "Надпись".


Рамка  - **Frame** - используется для оформления, а также для группировки переключателей. Сверху на рамке можно сделать надпись с помощью свойства **Caption**. Если нужно создать элемент внутри рамки, то перед его рисованием рамку выделяют, тогда рамка может служить контейнером для группы переключателей.

Кнопка  **Command Button** - в основном используется для инициирования выполнения некоторых действий, вызываемых нажатием кнопки, например запуск программы или остановка ее выполнения, печать результатов и т.д.

Свойства:

Caption	Возвращает текст, отображаемый на кнопке
Cancel	Допустимые значения: True (устанавливаются отменяющие функции для кнопки, т.е. нажатие клавиши <Esc> приводит к тем же результатам, что и нажатие кнопки) и False (в противном случае)
Visible	Допустимые значения: True (кнопка отображается во время выполнения программы) и False (в противном случае)
Enabled	Допустимые значения: True (разрешено нажатие кнопки пользователем) и False (в противном случае)
Picture	Внедряет на поверхность кнопки картинку. Например, <code>CommandButton1.Picture = _</code>

<p>LoadPicture("c:\my_doc\круг.bmp") Функция LoadPicture(ПолноеИмяФайла) считывает графическое изображение. Аргумент ПолноеИмяФайла указывает полное имя графического файла</p>
--

Переключатель  - **OptionButton** - позволяет пользователю выбрать один из нескольких взаимоисключающих параметров или действий. Переключатели обычно отображаются группами, обеспечивая возможность выбора альтернативного варианта.

Наиболее часто используемые свойства:

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)
Capture	Надпись, отображаемая рядом с переключателем







Флажок  **CheckBox** - представляет пользователю возможность выбора. Флажок обычно имеет два состояния: установленное () и сброшенное (). Но может настраиваться на выбор из трех альтернатив. Флажок имеет те же свойства Value, Enabled, Visible, Capture, что и переключатель. Кроме того флажок обладает уникальным свойством TripleState, позволяющим производить выбор из трех альтернатив. Свойство TripleState может принимать два значения: False (выбор из двух альтернатив), и True (выбор из трех альтернатив True, False и Null).

Рисунок  **Image** - используется для отображения графических файлов в формате bmp, cur, gif, ico, jpg, wmf. Основные свойства:


AutoSize	Принимает логические значения и устанавливает, должен ли объект автоматически изменять размер для того, чтобы поместить изображение целиком
Picture	Задает отображаемый графический файл. Используется с функцией LoadPicture
PictureSizeMode	Устанавливает масштабирование рисунка.

	<p>Допустимые значения:</p> <ul style="list-style-type: none"> • <code>fmPictureSizeModeClip</code> (не помещающиеся в границах объекта части рисунка обрезаются); • <code>fmPictureSizeModeStretch</code> (рисунок масштабируется так, чтобы он занимал всю поверхность объекта); • <code>fmPictureSizeModeZoom</code> (рисунок масштабируется с сохранением относительных размеров так, чтобы он помещался целиком внутри объекта)
<code>PictureAlignment</code>	<p>Устанавливает расположение изображения внутри элемента управления. Допустимые значения:</p> <ul style="list-style-type: none"> • <code>fmPictureAlignmentTopLeft</code> (в левом верхнем углу); • <code>fmPictureAlignmentTopRight</code> (в правом верхнем углу); • <code>fmPictureAlignmentTopCenter</code> (в центре); • <code>fmPictureAlignmentBottomLeft</code> (в левом нижнем углу); • <code>fmPictureAlignmentBottomRight</code> (в правом нижнем углу)

Поле со списком  `ComboBox`. В это поле пользователь может вводить текст так же, как и в `TextBox`, а кроме этого, если нажать , то откроется список, из которого можно выбрать нужную строку.

Свойства:

- **Text** - содержимое строки, введенное пользователем или выбранное из списка.
- **List** - строки списка (многострочное свойство).
- **ListIndex** - номер выбранной пользователем строки (нумерация начинается с нуля, если никакая строка не была выбрана, то свойство равно -1).

Список  `ListBox` - применяется для хранения списка значений. Из списка пользователь может выбрать одно или несколько

значений, которые в последующем будут использоваться в тексте программы.

Наиболее часто используемые свойства:

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
Text Column	Устанавливает столбец в списке, элемент которого возвращается свойством text
Text	Возвращает выбранный в списке элемент
List	Возвращает элемент списка, стоящий на пересечении указанных строки и столбца, синтаксис: List (row, column)
RowSource	Устанавливает диапазон, содержащий элементы списка
Multiselect	Устанавливает способ выбора элементов списка. Допустимые значения: <ul style="list-style-type: none"> ▪ 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	Возвращает массив значений типа Boolean для списка, который допускает множественный выбор. Каждый элемент массива содержит по одному элементу, соответствующему каждому элементу списка. Если значение элемента в массиве Selected равно True, то соответствующий элемент списка выбран. Допустимые значения: True (если элемент выбран) и False (в противном случае), используется для определения выделенного текста, когда свойство Multiselect имеет значение fmMultiSelectMulti или fmMultiSelectExtended
ColumnWidths	ColumnWidths = string, где string - строка,

	<p>устанавливающая ширину столбцов. В примере устанавливается ширина каждого из трех столбцов списка: With ListBox1 .ColumnCount = 3 .ColumnWidths = "20;30;30" End With</p>
ColumnHeads	<p>допустимые значения: true (выводятся заголовки столбцов раскрывающегося списка) и false (в противном случае)</p>
ListStyle	<p>допустимые значения:</p> <ul style="list-style-type: none"> ▪ fmListStylePlain – выбранный элемент из списка выделяется цветом; ▪ fmListStyleOption – перед каждым элементом в списке располагается флажок, и выбор элемента из списка соответствует установке этого флажка

Заполнить список можно одним из следующих способов:

Поэлементно, если список состоит из одной колонки	<pre>With ListBox1 .AddItem "ListBox" .AddItem "ComboBox" .AddItem "OptionButton" .ListIndex=0 End With</pre>
Массивом, если список состоит из одной колонки	<pre>With ListBox1 .List Array(1,2,3,4,5,6,7,8) .ListIndex=1 End With</pre>
Из диапазона A1:B4, в который предварительно введены элементы списка.	<pre>With ListBox1 .ColumnCount = 2 .RowSource = ("A1:B4") End With</pre>
Поэлементно, если список состоит из нескольких колонок, например, двух	<pre>With ListBox1 .ColumnCount = 2 .List(0,0)="Иванова" .List(0,1)="Мария" .List(1,0)="Петров" .List(1,1)="Алексей" End With</pre>

Массивом, если список состоит из нескольких колонок, например, двух	<pre>With ListBox1 .ColumnCount = 2 .List = a End With</pre>
---	--

Наиболее часто используемые методы элемента управления `ListBox`:

`Clear` – удаляет все элементы из списка;

`RemoveItem (index)` – удаляет из списка указанный элемент, где `index` – номер элемента;

`AddItem ([item [,varIndex]])` – добавляет элемент в список, где `item` – элемент (строковое выражение), добавляемый в список, и `varIndex` – номер добавляемого элемента.

Пользовательская форма `UserForm`

Пользовательская форма `UserForm` предоставляет пользователю возможность создавать диалоговые окна создаваемых приложений. Она служит базой для пользовательского диалогового окна, на котором в зависимости от решаемой задачи размещаются требуемые элементы управления.

Семейство `UserForms` является семейством, компоненты которого представляют все загруженные формы `UserForm` в приложении. Как и все семейства, `UserForms` имеет свойства: `Count` (возвращает число компонентов в семействе), `Item` (возвращает определенный компонент семейства), `Add` (добавляет к семейству новый компонент).

Наиболее часто используемые свойства объекта `UserForm`:

<code>Name</code>	Возвращает имя пользовательской формы
<code>Caption</code>	Возвращает текст, отображаемый в строке заголовка формы
<code>BackColor</code>	Возвращает цвет фона формы
<code>BorderStyle</code>	Устанавливает тип границы
<code>Picture</code>	Указывает рисунок, отображаемый как фон формы

Наиболее часто используемые методы объекта `UserForm`:

<code>Show</code>	Отображает форму на экране
<code>Hide</code>	Закрывает форму
<code>Move</code>	Изменяет положение и размер формы
<code>PrintForm</code>	Печатает изображение формы

Основные события объекта UserForm:

Activate	Иницируется всякий раз, когда окно формы становится активным
Initialize	Иницируется всякий раз, когда форма впервые загружается в память посредством выполнения оператора Load или с помощью метода Show
Resize	Иницируется при изменении размеров формы
Terminate	Иницируется всякий раз, когда форма выгружается из памяти

2. Типы данных, операции, встроенные функции

Тип данных определяет множество допустимых значений, которые может принимать указанная переменная. В VBA имеются следующие основные типы данных:

Типы данных	Размер (байт)	Диапазон значений
Byte (байт)	1	От 1 до 255
Boolean (логический)	2	True или False
Integer (целое число)	2	От -32 768 до 32 767
Long (длинное целое число)	4	От -2 147 483 648 до 2 147 483 647
Single (число с плавающей запятой обычной точности)	4	От -3,402823E38 до -1,401298E-45 для отрицательных значений; от 1,401298E-45 до 3,402823E38 для положительных значений
Double (число с плавающей запятой двойной точности)	8	От -1,7976931346232E308 до -4,9406565841247E-324 для отрицательных значений; от 4,9406565841247E-324 до 1,7976931346232E308 для положительных значений
Currency (денежный)	8	От -992 337 203 685 477,5808 до 992 337 203 685 477,5807
Object (объект)	4	Любой указатель объекта
String (строка переменной длины)	10+длина строки	От 0 до приблизительно 2 миллиардов
Variant (вариант)		Может использоваться для

		хранения различных типов данных: даты/времени, чисел с плавающей точкой, целых чисел, строк, объектов. Данные типа Variant могут иметь особое значение Null, которое означает, что данные отсутствуют, неизвестны или неприменимы.
--	--	--

Если переменная не объявлена, то по умолчанию имеет тип **Variant**.

Ссылки на объекты

Кроме обычных переменных, в VBA часто используются переменные, представляющие собой ссылку на объект. Использование переменной-объекта немного отличается от применения обычных переменных: нужно не только объявить такую переменную, но и перед ее использованием назначить соответствующий объект с помощью специального оператора Set. Синтаксис объявления и назначения следующий:

```
Dim <имяПеременной> As Object
```

```
Set <имяПеременной> = <ссылкаНаОбъект>
```

Объектная переменная будет указывать на объект до тех пор, пока мы другим оператором Set не присвоим ей ссылку на другой объект этого же типа или значение Nothing, означающее, что переменная не содержит никакой ссылки, например:

```
Set txt = Nothing
```

Объектные переменные в отличие от обычных переменных, содержащих значения, содержат только ссылки на объекты, а не сами объекты или их копии.

Описание переменных

Синтаксис:

```
Dim [WithEvents] ИмяПеременной [[[Индексы]]] [As [New] Тип] _  
[, [WithEvents] ИмяПеременной [[[Индексы]]] [As [New] Тип]] ...
```

Аргументы:

WithEvents	Ключевое слово, указывающее, что аргумент ИмяПеременной является именем объектной переменной, которая используется при отклике на
------------	---

	события, генерируемые объектом ActiveX (т.е. который может быть открыт для других приложений и средств программирования)
ИмяПеременной	Имя переменной
Индексы	Размерности переменной массива; допускается описание до 60 размерностей. Для задания аргумента Индексы используется следующий синтаксис: [Нижний To] Верхний [, [Нижний To] Верхний] ... Если нижний индекс не задан явно, нижняя граница массива определяется инструкцией Option Base. Если отсутствует инструкция Option Base, нижняя граница массива равна нулю.
New	Ключевое слово, включающее возможность неявного создания объекта. Если указано ключевое слово New при описании объектной переменной, новый экземпляр объекта создается при первой ссылке на него.
Тип	Тип данных переменной

Переменные, описанные с помощью ключевого слова Dim на уровне модуля, доступны для всех процедур в данном модуле. Переменные, описанные на уровне процедуры, доступны только в данной процедуре.

Допустимые имена. В VBA имеются следующие ограничения на имена переменных, функций, процедур, констант и других объектов:

1. Длина имени не должна превышать 255 символов.
2. Имя не может содержать точек, пробелов и следующих символов: %, !, &, #, @, \$.
3. Имя может содержать любую комбинацию букв, цифр, и символов, начинающуюся с буквы.
4. Имена должны быть уникальны в области, в которой они определены.
5. Не следует использовать имена, совпадающие с ключевыми словами VBA и именами встроенных процедур и функций.

Любую переменную, которая должна быть доступна для всех процедур (функций и подпрограмм), используемых объектами формы, нужно объявлять в области (General) модуля.

Константы. В отличие от переменных не могут изменять свои значения. Синтаксис:

[Public | Private] Const ИмяКонстанты [As тип] = Выражение

Аргументы:

Public	Ключевое слово, используемое на уровне модуля для описания констант, доступных всем процедурам во всех модулях. Не допускается в процедурах.
Private	Ключевое слово, используемое на уровне модуля для описания констант, доступных только внутри модуля, в котором выполняется описание. Не допускается в процедурах.
ИмяКонстанты	Имя константы
тип	Один из поддерживаемых типов данных.
Выражение	Литерал, другая константа или любое сочетание, которое включает все арифметические или логические операторы, за исключением Is.

Операции VBA

Имеются три основных типа операций:

- Математические операции выполняются над числами. Их результатом являются числа.
- Операции отношения применяются не только к числам. Их результатом являются логические значения.
- Логические операции используются в логических выражениях. Их результатом являются логические значения.

Математические операции.

[Операнд1] + [Операнд2]	Сложение
[Операнд1] - [Операнд2]	Вычитание
-[Операнд]	Перемена знака
[Операнд1] * [Операнд2]	Умножение
[Операнд1] / [Операнд2]	Деление
[Операнд1] \ [Операнд2]	Целочисленное деление
[Операнд1] Mod [Операнд2]	Остаток от деления по модулю
[Операнд1] ^ [Операнд2]	Возведение в степень

Операции отношения

[Операнд1] < [Операнд2]	Меньше
[Операнд1] > [Операнд2]	Больше
[Операнд1] <= [Операнд2]	Меньше или равно
[Операнд1] >= [Операнд2]	Больше или равно
[Операнд1] <> [Операнд2]	Не равно
[Операнд1] = [Операнд2]	Равно
[Операнд1] Is [Операнд2]	Сравнение двух операндов, содержащих ссылки на объекты
[Операнд1] Like [Операнд2]	Сравнение двух строковых выражений

Логические операции

[Операнд1] And [Операнд2]	Логическое умножение
[Операнд1] Or [Операнд2]	Логическое сложение
[Операнд1] Xor [Операнд2]	Исключающее Or (или)
[Операнд1] Not [Операнд2]	Логическое отрицание

VBA выполняет операции в соответствии с их приоритетами.

Приоритет	Операция
1	Вызов функции и операции в скобках
2	^
3	- (смена знака)
4	*, /
5	\
6	Mod
7	+, -
8	>, <, >=, <=, <>, =
9	And
10	Or
11	Xor
12	Not

Математические функции

Функция	Возвращаемое значение
Abs(число)	Модуль (абсолютная величина) числа
Atn(число)	Арктангенс

Cos(число)	Косинус
Exp(число)	Экспонента, т.е. результат возведения основания натурального логарифма в указанную степень
Log(число)	Натуральный логарифм
Rnd(число)	Случайное число из интервала [0,1]
Sgn(число)	Знак числа
Sin(число)	Синус
Sqr(число)	Квадратный корень из числа
Tan(число)	Тангенс
Fix(число) и Int(число)	Обе функции отбрасывают дробную часть числа и возвращают целое значение. Различие: для отрицательного значения аргумента число функция Int возвращает ближайшее отрицательное целое число, меньшее либо равное указанному, а Fix - ближайшее отрицательное целое число, большее либо равное указанному

Функции проверки типов

Функции проверки типов проверяют, является ли переменная выражением специфицированного типа.

IsArray(переменная)	Является ли переменная массивом
IsDate(переменная)	Является ли переменная датой
IsEmpty(переменная)	Была ли переменная описана инструкцией Dim
IsError(переменная)	Является ли переменная кодом ошибки
IsNull(переменная)	Является ли переменная пустым значением (Null)
IsNumeric(переменная)	Является ли переменная числовым значением
IsObject(переменная)	Является ли переменная объектом

Функции преобразования типов

Функция	Возвращает действие	Возвращаемый тип
Str(N)	Возвращает строку, эквивалентную численному выражению N. В качестве допустимого десятичного разделителя	String

	воспринимает только точку. При наличии другого десятичного разделителя (например, запятой) для преобразования чисел в строки следует использовать функцию CStr	
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 S в число, то функция Val возвращает 0	Variant
Asc(S)	Возвращает число кода символа, соответствующее первой букве строки S. Например, буква «А» имеет код 65	Integer
Chr(N)	Возвращает строку из одного символа, соответствующего коду символа N, который должен быть числом между 0 и 255. Код символа 66 возвращает букву “В”	Символ

Чтобы представить числовое значение как дату, время, денежное значение или в специальном формате, следует использовать функцию Format, которая возвращает значение типа Variant (String), содержащее выражение, отформатированное согласно инструкциям, заданным в описании формата.

Синтаксис:

Format (Выражение [,Формат])

- Выражение - любое допустимое выражение
- Формат - любое допустимое именованное или определенное пользователем выражение формата.

Примером именованного формата является Fixed – формат действительного числа с двумя значащими цифрами после десятичной точки. При построении пользовательского формата возможно использование следующих символов:

- 0 – резервирует позицию цифрового разряда. Отображает цифру или ноль;

- # - резервирует позицию цифрового разряда. Отображает цифру или ничего не отображает;
- . - резервирует позицию десятичного разделителя;
- % - резервирует процентное отображение числа;
- : - разделитель часов, минут и секунд в категории форматов Time;
- / - разделитель дня, месяца и года в категории форматов Date;
- E+, E-, e+, e- - разделитель мантиссы и порядка в экспоненциальном формате.

Кроме функций Val и Str в VBA имеются следующие функции преобразования типов выражений из данного в указанный:

Функция	Тип, в который преобразуется выражение
CBool(выражение)	Boolean
CByte(выражение)	Byte
CCur(выражение)	Currency
CDate(выражение)	Date
CDbl(выражение)	Double
CDec(выражение)	Decimal
CInt(выражение)	Integer
CLng(выражение)	Long
CSng(выражение)	Single
CVar(выражение)	Variant
CStr(выражение)	String

Встроенные диалоговые окна

В проектах VBA часто встречаются две разновидности диалоговых окон: окна сообщений и окна ввода. Они встроены в VBA. Окно сообщений (MsgBox) выводит простейшие сообщения для пользователя, а окно ввода (InputBox) обеспечивает ввод информации.

Окно сообщения

Процедура MsgBox() выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия кнопки пользователем. Синтаксис создания окна сообщений:

MsgBox (Сообщение [,Атрибуты] [,Заголовок])

Сообщение — текст, отображаемый в окне сообщений, является обязательным аргументом. Эта строка должна быть заключена в двойные кавычки.

Атрибуты определяют особенности окна, т.е. различные кнопки и значки, отображаемые в нем. Аргумент Атрибуты позволяет управлять следующими параметрами окна сообщения: количеством кнопок в окне, типом кнопок и их размещением в окне, пиктограммой, отображаемой в окне.

Этот аргумент является целым числом и для достижения желаемого результата может быть представлен как сумма двух слагаемых:





Атрибуты = Параметр1 + Параметр2 .

Значение Параметр1 устанавливает число и тип кнопок в окне сообщений. В таблице 2.1 приведены возможные значения этого параметра:

Таблица 2.1.

Параметр1	Набор кнопок
0	Отображается только кнопка "ОК"
1	Отображаются кнопки "ОК" и "Отмена"
2	Отображаются кнопки "Прервать", "Повтор", "Пропустить"
3	Отображаются кнопки "Да", "Нет", "Отмена"
4	Отображаются кнопки "Да", "Нет"
5	Отображаются кнопки "Повтор", "Отмена"

Таблица 2.2.

Параметр2	Вид сообщения	Пиктограмма
16	Отображается значок критического сообщения	
32	Отображается вопросительный знак (предупреждение)	
48	Отображается восклицательный знак (предупреждение)	
64	Отображается знак информационного сообщения	

Если аргумент Атрибуты не указан, то VBA предполагает, что в диалоговом окне сообщений присутствует только кнопка "ОК".

Значение Параметр2 определяет вид сообщения и пиктограмму, которая помещается в окне сообщений (таблица 2.2).

Заголовок — строка в заголовке окна сообщений. Если этот аргумент опущен, то в строке заголовка отображается "Microsoft Excel".

Окно ввода

Функция InputBox() применяется для ввода чисел или текста, выводит на экран диалоговое окно, содержащее сообщение, поле ввода и две кнопки **ОК** и **Cancel**. Устанавливает режим ожидания ввода текста пользователем и нажатия кнопки, а затем, при нажатии кнопки **ОК**, возвращает значение типа String, содержащее текст, введенный в поле ввода. При нажатии кнопки **Cancel** возвращает пустую строку.

Синтаксис этой функции:

InputBox(Сообщение[, Заголовок] [, Умолчание])

Сообщение — единственный обязательный аргумент; он служит подсказкой пользователю, какую информацию необходимо ввести в поле ввода.

Заголовок — это надпись в строке заголовка окна ввода.

Умолчание — значение, которое будет отображаться в поле ввода по умолчанию, пока пользователь не введет свое значение. Если этот аргумент опустить, то поле ввода отображается пустым.

Возвращаемым значением данной функции является информация, вводимая пользователем. Возвращаемое значение можно использовать в окнах сообщений, поместить в ячейку рабочего листа, применить в вычислениях и т.д.

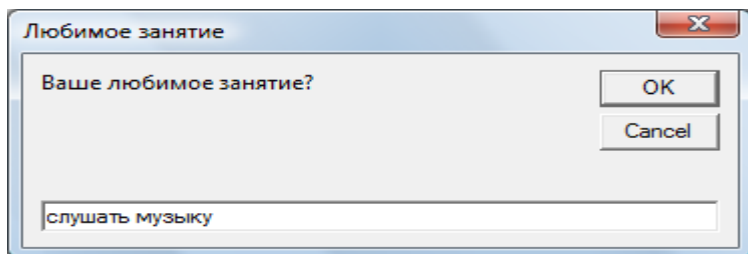


Рис. 2.1. Окно ввода

Следующий код отображает на экране окно ввода, показанное на рис. 2.1. После ввода ответа и нажатия **ОК** отображается окно сообщения, показанное на рис. 2.2.

```
Private Sub CommandButton1_Click()  
Dim s As String  
s = InputBox("Ваше любимое занятие? ", "Любимое занятие")  
MsgBox "Мое любимое занятие - " & s, 1, "Ответ"  
End Sub
```

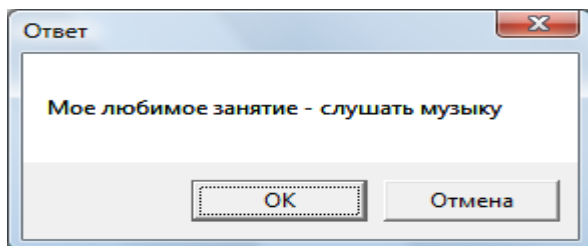


Рис. 2.2 Окно сообщения

3. Управляющие конструкции VBA

Как и во всех других языках программирования, в VBA имеются различные управляющие конструкции, позволяющие изменять порядок выполнения алгоритма и программы. Если в алгоритме и программе нет управляющих конструкций, то происходит последовательное выполнение операторов (линейный алгоритм). В самых простых случаях этого бывает достаточно, однако обычно все-таки бывает необходимо изменять порядок выполнения операторов при выполнении определенных условий, либо пропускать выполнение некоторых операторов, либо, наоборот, многократно повторять их. Для реализации любых алгоритмов достаточно иметь два вида конструкций управления: циклы и ветвления.

3.1. Оператор присваивания

Оператор присваивания присваивает значение выражения переменной, константе или свойству объекта. Оператор присваивания всегда включает знак равенства (=).

Синтаксис:

[Let] Переменная (или Постоянная или Свойство объекта) =
Выражение

Оператор присваивания предписывает вычислить выражение, заданное в его правой части, и присвоить результат переменной, имя которой указано в левой части.

Для присвоения переменной ссылки на объект применяется инструкция `Set`. В следующем примере инструкция `Set` присваивает переменной `Область` диапазон `A1:B3`:

```
Set Область = Range("A1:B3")
```

В общем случае инструкция `Set` имеет следующий синтаксис:

```
Set ОбъектнаяПеременная {[New] ОбъектноеВыражение |  
Nothing}
```

- ключевое слово `New` используется при создании нового класса;
- ключевое слово `Nothing` позволяет освободить все системные ресурсы и ресурсы памяти, выделенные для объекта, на который имелась ссылка (она удаляет объект из памяти).

Примечания

Оператор With. Оператор `With` избавляет программиста использовать большое количество повторений имени одного и того же объекта при работе с его свойствами и методами. Кроме того он структурирует код, делая его более прозрачным:

```
With Range("A1")  
    .Value = 3  
    .Font.Italic = True  
End With
```

Допустимо также использование вложенных операторов `With`.

Перенос строки. Расположение символов {Пробел}+{Знак подчеркивания} в конце строки обеспечивает то, что последующая строка является продолжением предыдущей.

Расположение нескольких операторов на одной строке.

Использование знака двоеточия позволяет разместить несколько операторов на одной строке. Следующие две конструкции эквивалентны:

```
x = x + 1  
y = x + 2  
и  
x = x + 1 : y = x + 2
```

Комментарии. Работая с программой, удобно использовать комментарии. В языке VBA существуют два способа ввода комментариев:

- Применение апострофа ('). Его можно ставить в любом месте строки. Все символы, начиная от апострофа до конца строки, будут восприниматься компилятором как комментарий.
- Применение зарезервированного слова Rem вместо апострофа.

3.2. Оператор безусловного перехода

Оператор безусловного перехода GoTo задает переход на указанную строку внутри процедуры. Обязательный параметр Строка может быть любой меткой строки или номером строки.

GoTo Строка

Для использования оператора безусловного перехода строке, куда должно быть передано управление, необходимо присвоить метку. Метка должна начинаться с буквы и заканчиваться двоеточием. Действие оператора GoTo состоит в передаче управления меченой строке.

3.3. Операторы ветвления

Управляющие конструкции ветвления (перехода) позволяют проверить некоторое условие и, в зависимости от результатов этой проверки, выполнить ту или иную группу операторов. Для организации ветвлений в VBA используются различные формы оператора ветвления If и оператор выбора Select Case.

Оператор ветвления

Краткая форма оператора ветвления If применяется для проверки условия, а затем либо выполнения, либо пропуска одного оператора или блока из нескольких операторов. Краткая форма оператора может иметь как однострочную, так и блочную форму.

В одну строку краткая форма оператора может быть записана так:
If <условие> Then <оператор>

В блочной форме краткая форма оператора выглядит следующим образом:

```
If <условие> Then  
    <оператор1>  
    <оператор2>
```

...

End if

Если условие принимает значение True, то выполняется оператор или блок операторов, если False, то оператор или блок операторов пропускается.

Полная форма оператора If применяется в тех случаях, когда есть два различных блока операторов, и по результатам проверки условия нужно выполнить один из них. Такая форма оператора If всегда имеет блочную форму записи.

If Условие Then

<блокОператоров1>

Else

<блокОператоров2>

End if

Если условие истинно, выполняется первый блок операторов, в противном случае – второй блок.

Пример 1. Создать программу решения уравнения $ax + b = 0$. Для ввода значений коэффициентов a и b на окне формы разместить два поля ввода TextBox, результат вывести на надпись (рис. 3.1).

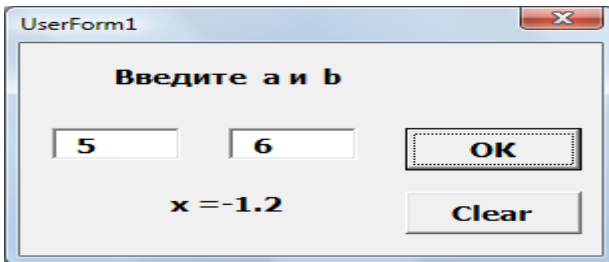


Рис. 3.1. Вид окна программы

Код модуля формы

```
Private Sub CommandButton1_Click()  
Dim a, b, x As Single  
Dim s As String  
a = TextBox1.Text  
b = TextBox2.Text  
If (a = 0) And (b <> 0) Then s = "нет решения"  
If (a = 0) And (b = 0) Then s = "бесконечное множество решений"  
If a <> 0 Then s = "x =" + (Str(Round(-b / a, 3)))  
Label2.Caption = s
```

```
End Sub
```

```
Private Sub CommandButton2_Click()
```

```
    TextBox1.Text = ""
```

```
    TextBox2.Text = ""
```

```
    Label2.Caption = ""
```

```
End Sub
```

Пример 2. Создать программу для умножения, сложения, вычитания, деления двух чисел и возведения в квадрат, извлечения квадратного корня из одного числа. Числа ввести с помощью полей ввода TextBox. Для управления видимостью второго поля ввода установить флажок CheckBox. При сброшенном флажке второе поле невидимо. На рабочем листе создать кнопочное меню - установить командные кнопки «Сложение», «Вычитание», «Умножение», «Деление», «Возведение в квадрат», «Извлечение квадратного корня», «Очистить». Результат вывести с помощью окна сообщений (рис. 3.2).

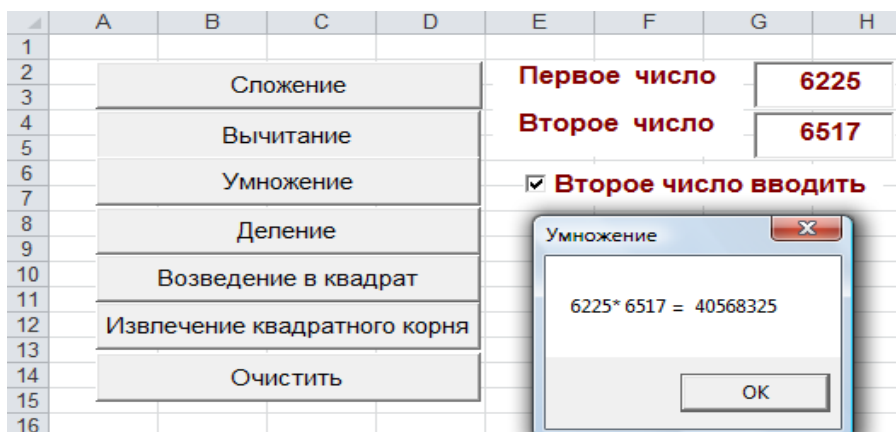


Рис. 3.2.

Код модуля рабочего листа Лист1

```
Dim a, b As Long
```

```
Private Sub CheckBox1_Change()
```

```
    If CheckBox1.Value Then
```

```
        TextBox2.Visible = True
```

```
    Else
```

```
        TextBox2.Visible = False
```

```
    End If
```

End Sub

Private Sub CommandButton1_Click()

a = Val(TextBox1.Text)

b = Val(TextBox2.Text)

MsgBox Str(a) + "+" + Str(b) + " = " + Str(a + b), , "Сложение"

End Sub

Private Sub CommandButton2_Click()

a = Val(TextBox1.Text)

b = Val(TextBox2.Text)

MsgBox Str(a) + "-" + Str(b) + " = " + Str(a - b), , "Вычитание"

End Sub

Private Sub CommandButton3_Click()

a = Val(TextBox1.Text)

b = Val(TextBox2.Text)

MsgBox Str(a) + "*" + Str(b) + " = " + Str(a * b), , "Умножение"

End Sub

Private Sub CommandButton4_Click()

a = Val(TextBox1.Text)

b = Val(TextBox2.Text)

If (b = 0) And (a <> 0) Then

MsgBox Str(a) + "/" + Str(b), 48, "Деление на ноль"

Else

If (b = 0) And (a = 0) Then

MsgBox Str(a) + "/" + Str(b), 16, "Деление ноля на ноль"

Else

MsgBox Str(a) + "/" + Str(b) + " = " + Str(a / b), , "Деление"

End If

End If

End Sub

Private Sub CommandButton5_Click()

a = Val(TextBox1.Text)

MsgBox Str(a) + "*" + Str(a) + " = " + Str(a * a), , "Возведение в квадрат"

End Sub

Private Sub CommandButton6_Click()


```

a = Val(TextBox1.Text)
If a < 0 Then
MsgBox Str(a), 48, "Введите положительное число"
Else
MsgBox "Квадратный корень из " + Str(a) + " = " & Sqr(a), , "Извлечение
квадратного корня"
End If
End Sub

Private Sub CommandButton7_Click()
TextBox1.Text = ""
TextBox2.Text = ""
End Sub

```

Оператор выбора

Иногда приходится делать выбор одного действия из целой группы действий на основе проверки нескольких различных условий. В таких случаях удобно использовать специально предназначенный для этого оператор выбора `Select Case`, имеющий следующий синтаксис:

```

Select Case <проверяемоеВыражение>
Case <списокЗначений1>
<блокОператоров1>
Case <списокЗначений2>
<блокОператоров2>
...
[Case Else
<блокОператоровElse>]
End Select

```

Здесь `select`, `case`, `of`, `else`, `end` - зарезервированные слова (выбрать, случай, из, иначе, конец).

Проверяемое выражение вычисляется в начале работы оператора `Select Case`. Это выражение может возвращать значение любого типа, например, логическое, числовое или строковое.

Список значений представляет собой одно или несколько выражений, разделенных запятой. При выполнении оператора проверяется, соответствует ли хотя бы один из элементов этого списка значению проверяемого выражения.

Если хотя бы один из элементов списка соответствует проверяемому выражению, то выполняется соответствующая группа операторов, и на этом выполнение оператора Select Case заканчивается, а остальные списки выражений не проверяются, т.е. отыскивается только первый подходящий элемент списка выражений. Если ни один из элементов всех этих списков не соответствует значению проверяемого выражения, выполняются операторы группы Else, если такая присутствует.

Пример 1. Создать программу, которая вводит два числа и знак арифметического действия и над двумя данными числами производит указанное арифметическое действие. Арифметическое действие выбрать из раскрывающегося списка ComboBox. Числа вводить с помощью TextBox. Результат действия с указанием операндов добавить в список ListBox (рис.3.3). Арифметические действия: сложение, вычитание, умножение, деление. Предусмотреть обработку ошибочных ситуаций. Сообщения об ошибках добавить в список ListBox.

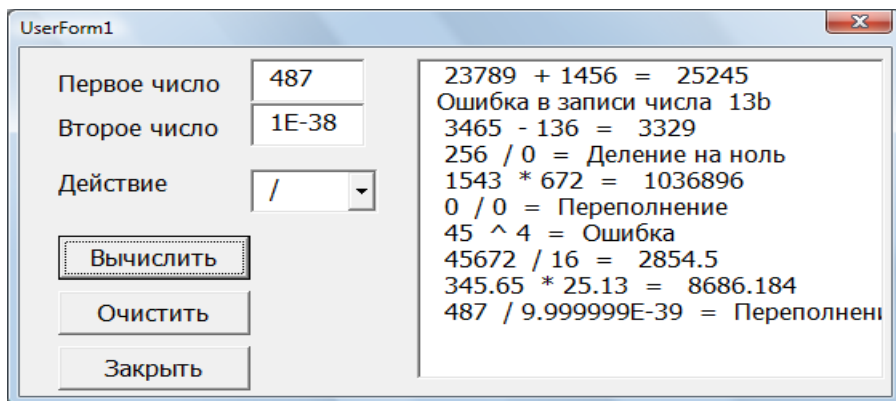


Рис. 3.3.

Для предотвращения возникновения ошибочных ситуаций в программе анализируются вводимые данные. При вводе некорректных данных т.е. данных, которые не могут быть преобразованы в число, сообщение об ошибке добавляется в список ListBox и осуществляется выход из процедуры.

Для безаварийного продолжения работы программы используется инструкция On Error, которая перехватывает ошибку и устанавливает,

что программа должна делать в случае возникновения ошибки. Один из синтаксисов использования этой инструкции:

On Error Goto строка

Активизирует обработчик ошибок, начало которого определяется обязательным аргументом строки, значением которого может быть либо метка строки, либо номер строки. В нашем примере при возникновении ошибки управление передается в строку, помеченную меткой Ошибка. Обработчик ошибок с помощью оператора выбора анализирует код ошибки. Обработчик ошибок обычно включает объект Err, который содержит информацию об ошибках выполнения. Свойство Number объекта Err возвращает код ошибки. Коды часто встречающихся ошибок приведены в таблице 6.1. Если при выполнении операции деления в поле Второе число ввести нулевое значение, возникает ошибка деления на нуль с кодом 11. Если при выполнении операции деления в поля Первое число и Второе число ввести нулевые значения или в поле Второе число ввести значение близкое к нулю, например, 1E-38, возникает ошибка переполнения с кодом 6. Обработчик ошибок соответствующее сообщение добавляет в список ListBox. При появлении других ошибок, выполнение программы прерывается с информированием пользователя об ошибке. Свойство Description объекта Err возвращает строковое выражение, содержащее текст сообщения об ошибке. Осуществляется выход из процедуры.

Примечание

Более подробно обработка ошибочных ситуаций рассматривается в пункте 6.3.

Код модуля формы

' Обработка события Initialize объекта Userform

```
Sub Userform_Initialize()
```

```
With ComboBox1
```

```
.List = Array("+", "-", "*", "/")
```

```
.ListIndex = -1
```

```
End With
```

```
End Sub
```

```
Private Sub CommandButton1_Click()
```

```
Dim x, y, z As Single
```

```
Dim d, s, s1 As String
```

```

Dim k As Integer
' передача управления на обработчик ошибок,
' помеченный меткой Ошибка
On Error GoTo Ошибка
' Проверка, являются ли введенные значения числами
If IsNumeric(TextBox1.Text) = False Then
s = "Ошибка в записи числа " + TextBox1.Text
ListBox1.AddItem (s)
TextBox1.SetFocus
Exit Sub
End If
If IsNumeric(TextBox2.Text) = False Then
s = "Ошибка в записи числа " + TextBox2.Text
ListBox1.AddItem (s)
TextBox2.SetFocus
Exit Sub
End If
x = CSng(TextBox1.Text)
y = CSng(TextBox2.Text)
d = ComboBox1.Text
s1 = "": k = 0
Select Case d
Case "+"
z = x + y
Case "-"
z = x - y
Case "*"
z = x * y
Case "/"
z = x / y
Case Else
s1 = "Ошибка": k = 1
End Select
If k = 0 Then
s = Str(x) + " " + d + Str(y) + " = " + Str(z)
Else
s = Str(x) + " " + d + Str(y) + " = " + s1

```

```

End If
ListBox1.AddItem (s)
' Выход из процедуры в случае
' успешного нахождения результата
Exit Sub
' Обработчик ошибок
Ошибка:
Select Case Err.Number
Case 11
ListBox1.AddItem (Str(x) + " " + d + Str(y) + " = " + "Деление на нуль")
Case 6
ListBox1.AddItem (Str(x) + " " + d + Str(y) + " = " + "Переполнение")
Case Else
MsgBox "Произошла ошибка: " & Err.Description
End Select
End Sub

Private Sub CommandButton2_Click()
    TextBox1.Text = ""
    TextBox2.Text = ""
    ComboBox1.ListIndex = -1
End Sub

Private Sub CommandButton3_Click()
    UserForm1.Hide
End Sub

```

Задания для самостоятельного выполнения.

1. Принадлежит ли точка $A(x,y)$ закрашенной области (рис. 3.5)? Для ввода координат точки и вывода результата создать диалоговое окно.

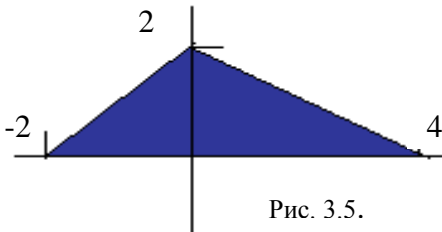


Рис. 3.5.

2. Создать программу для вычисления площадей геометрических фигур: прямоугольника, квадрата, трапеции, треугольника по трем сторонам, круга и кольца. Для ввода исходных данных на рабочем листе установить три поля ввода TextBox. Для управления видимостью второго и третьего полей ввода установить два флажка CheckBox. При установленных флажках второе и третье поля невидимы. На рабочем листе создать кнопочное меню - установить командные кнопки «Площадь прямоугольника», «Площадь квадрата», «Площадь трапеции», «Площадь треугольника», «Площадь круга», «Площадь кольца», «Очистить». Результат вывести с помощью окна сообщений (рис. 3.6).

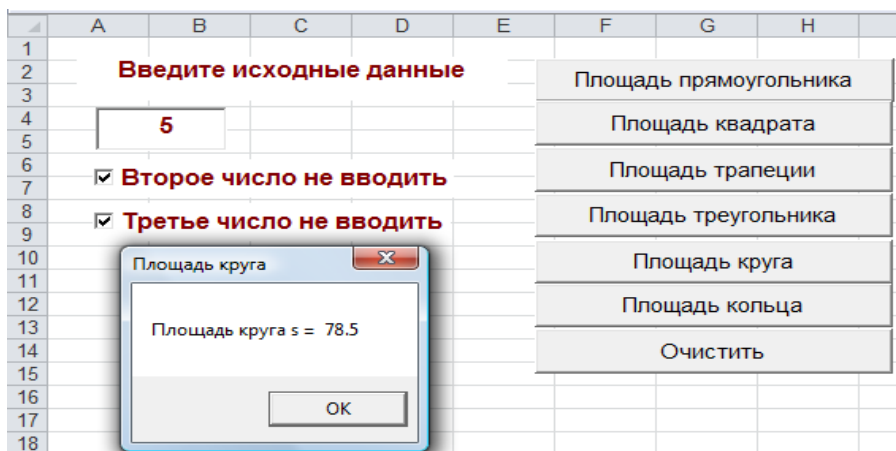


Рис. 3.6.

3. Создать программу нахождения числа дней в месяце, если даны: номер месяца n – целое число от 1 до 12 и год. Для ввода года, номера месяца и вывода результата на окне формы разместите три поля ввода TextBox, для управления работой программы – три командные кнопки.

4. По номеру месяца вывести сообщение о времени года. Номер месяца ввести с помощью TextBox. Результат вывести на надпись и в зависимости от времени года в элемент управления Image1 загрузить разные рисунки. Рисунок можно загрузить используя свойство Picture и функцию LoadPicture: `Image1.Picture = LoadPicture("путь\имя файла")` (рис.3.7). Предусмотреть обработку ошибочных ситуаций. Если имя файла или путь к файлу заданы неверно, вывести об этом сообщение.

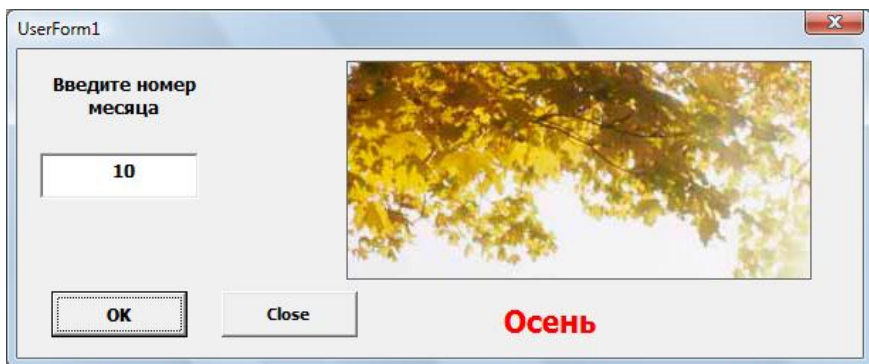


Рис. 3.7.

5. Создать программу, которая составляет предложения из слов, заранее введенных в три раскрывающихся списка ComboBox. В каждом списке не менее десяти слов. Одно слово берется из первого списка, одно – из второго, одно – из третьего. В первый список занесены названия зверей и птиц, во второй – наречия, характеризующие образ действия, в третий – глаголы, описывающие разные действия. Из каждого списка слово выбирается случайным образом. Для опеределения индекса случайно выбираемого элемента списка используется функция Rnd. Полученное с помощью функции Rnd случайное число умножается на число элементов списка и округляется до целого с помощью функции Int. Число элементов списка определяется свойством ListCount. Составленные предложения добавляются в список ListBox. (рис. 3.8).

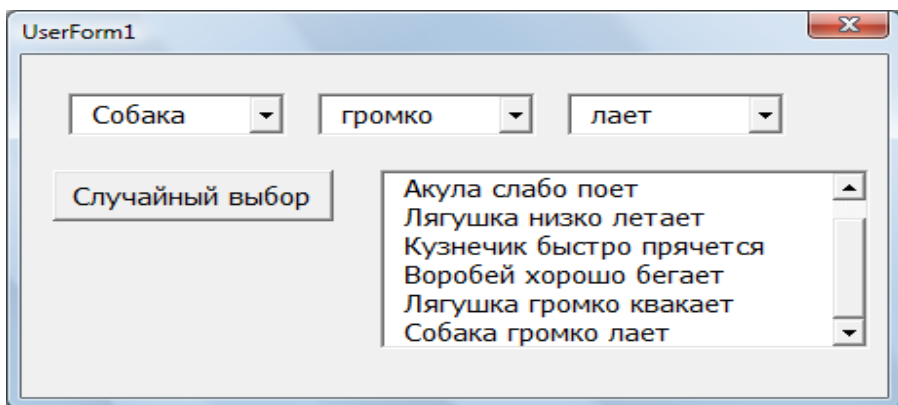


Рис. 3.8.

3.4. Операторы цикла

Циклы, в которых число повторений заранее известно, называются **арифметическими циклами**. Циклы, в которых известно только условие завершения цикла, а число повторений цикла заранее неизвестно, называются **итерационными циклами**.

В VBA есть богатый выбор средств организации циклов, которые можно разделить на две основные группы:

1. *Циклы с перечислением For ... Next*
2. *Циклы с условием Do ... Loop*

Цикл For ... Next используется для организации арифметических циклов.

Циклы типа Do ... Loop используются в тех случаях, когда заранее неизвестно сколько раз должно быть повторено выполнение блока операторов, составляющего тело цикла. Такой цикл продолжает свою работу до тех пор, пока не будет выполнено определенное условие. Такие циклы делятся на:

1. *Циклы с предусловием Do While ... Loop*
2. *Циклы с постусловием Do Until ... Loop*

Цикл For ... Next

Синтаксис:

```
For <счетчик> = <начальноеЗначение> To <конечноеЗначение>  
[Step <шаг>]  
<блокОператоров>  
[Exit For]  
[<блокОператоров>]  
Next [<счетчик>]
```

В этой конструкции цикла:

- <шаг> может быть положительным, так и отрицательным числом. При отрицательном шаге конечное значение должно быть меньше начального значения либо равно ему.
- После завершения работы цикла For ... Next переменная, которая использовалась в качестве счетчика, получает значение, обязательно превосходящее конечное значение в том случае, если шаг положителен, и меньшее конечного значения, если шаг отрицателен.
- Если начальное и конечное значения совпадают, тело цикла выполняется один раз.

- Exit For – альтернативный способ выхода из цикла.

Порядок выполнения оператора: повторяет выполнение блока операторов, пока счетчик изменяется от начального значения до конечного с указанным шагом. Если шаг не указан, то он полагается равным 1.

Пример 1. Вычислить сумму $S = \frac{tg^2 1}{1^2 + 1} + \frac{tg^2 2}{2^2 + 1} + \dots + \frac{tg^2 k}{k^2 + 1}$.

Параметр цикла i (тип integer) принимает значения от 1 до k . Каждое слагаемое вычисляется по формуле: $\frac{tg^2 i}{i^2 + 1}$. Значение k ввести с помощью поля ввода TextBox, значение суммы вывести на надпись (рис.3.9).

Код модуля формы

```
Private Sub CommandButton1_Click()
Dim s As Single, i, k As Integer
k = Val(TextBox1.Text)
' При вводе отрицательного значения k
' осуществляется выход из процедуры
If k < 0 Then
MsgBox "Введите положительное число", 48
Exit Sub
End If
s = 0
For i = 1 To k Step 1
s = s + Tan(i) ^ 2 / (i ^ 2 + 1)
Next i
Label2.Caption = "S = " & s
End Sub

Private Sub CommandButton2_Click()
UserForm1.Hide
End Sub

Private Sub CommandButton3_Click()
TextBox1.Text = ""
Label1.Caption = ""
```

End Sub

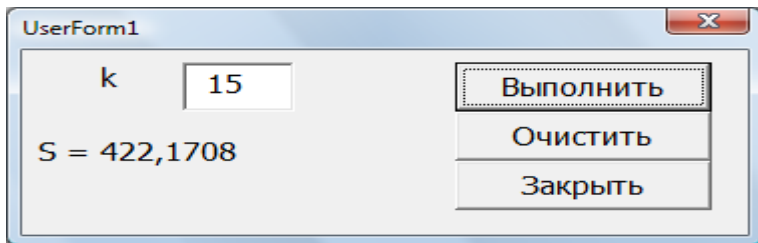


Рис. 3.9. Вид окна программы

Пример 2. Протабулировать функцию $y = 3x - 4\ln x - 5$ на отрезке $[a, b]$ с шагом h . Для ввода значений a , b , h на рабочем листе установить три поля ввода `TextBox`, для вывода значений аргумента и функции – список `ListBox`, для управления работой программы – две командные кнопки (рис. 3.10).

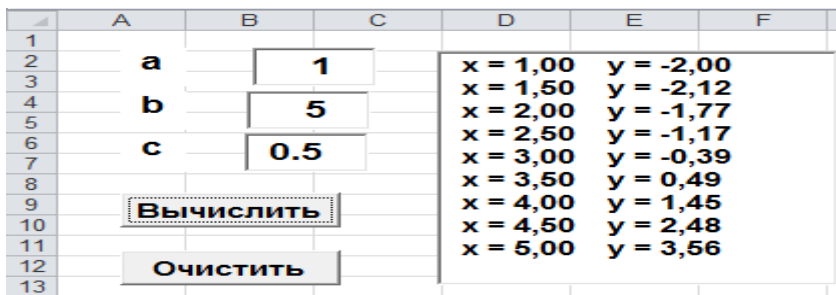


Рис. 3.10.

Код модуля рабочего листа Лист1

```
Private Sub CommandButton1_Click()  
Dim x, y, a, b, h As Single, s As String  
s = ""  
a = Val(TextBox1.Text)  
b = Val(TextBox2.Text)  
h = Val(TextBox3.Text)  
' Проверка корректности исходных данных,  
' если введены некорректные данные,  
' осуществляется выход из процедуры  
If (h > 0) And (a > b) Then  
MsgBox "При положительном шаге a <= b", 48  
Exit Sub
```

```

End If
If (h < 0) And (a < b) Then
MsgBox "При отрицательном шаге a >= b", 48
Exit Sub
End If
If (a>0) And (b>0) Then
For x = a To b Step h
    y = 3 * x - 4 * Log(x) - 5
    ' Значения x и y выводятся с двумя знаками после запятой
    s = "x = " + Format(x, "#####0.00") + "    y = " + Format(y, "#####0.00")
    ListBox1.AddItem (s)
Next
Else
    MsgBox " Данное значение x не входит в область определения
функции lnх"
End If
End Sub

Private Sub CommandButton2_Click()
    TextBox1.Text = ""
    TextBox2.Text = ""
    TextBox3.Text = ""
    ListBox1.Clear
End Sub

```

Оператор цикла с предусловием – Do while

Используется для организации итерационных циклов.

Синтаксис:

```

Do while <условие>
<блокОператоров>
[Exit Do]
[<блокОператоров>]
Loop

```

Порядок выполнения оператора:

Проверяется условие, если условие выполняется, то выполняется блок операторов тела цикла и снова происходит возврат к проверке условия. Так повторяется, пока условие повторения цикла не примет

значение false. Как только значение условия становится false, оператор Do while завершает свою работу.

Если с самого начала условие не выполняется, то блок операторов тела цикла ни разу не выполнится.

Чтобы не произошло «зацикливания» в теле цикла величины, входящие в условие, должны меняться.

Альтернативный способ выхода из цикла представляет инструкция Exit Do.

Пример. Найти произведение ненулевых цифр натурального числа n. Для ввода значения n на окне формы установить поле ввода TextBox, для вывода значения произведения - надпись (рис. 3.11).

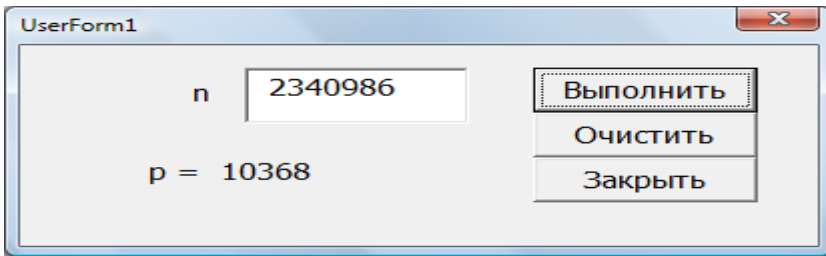


Рис. 3.11. Вид окна программы

Код модуля формы

```
Private Sub CommandButton1_Click()  
Dim n As Long, p, a As Integer  
n = Val(TextBox1.Text)  
' При вводе отрицательного значения n  
' осуществляется выход из процедуры  
If n < 0 Then  
MsgBox "Введите положительное число", 48  
Exit Sub  
End If  
p = 1  
Do While n > 0  
a = n Mod 10  
n = n \ 10  
If a <> 0 Then p = p * a  
Loop  
Label2.Caption = "p = " & p
```

```

End Sub

Private Sub CommandButton2_Click()
UserForm1.Hide
End Sub

Private Sub CommandButton3_Click()
TextBox1.Text = ""
Label2.Caption = ""
End Sub

```

Оператор цикла – Do until

Этот оператор используется в том случае, когда число повторений цикла заранее не известно.

Синтаксис:

```

Do until <условие>
<блокОператоров>
[Exit Do]
[<блокОператоров>]
Loop

```

Порядок выполнения оператора:

Выполнение оператора начинается с выполнения, блока операторов, затем проверяется условие. Если условие не выполняется, то происходит возврат к выполнению блока операторов. Так повторяется, пока значение условия остается ложным. Как только значение условия становится true, оператор завершает свою работу.

Если условие имеет значение true с самого начала, блок операторов тела цикла выполнится один раз. Чтобы оператор do until выполнялся конечное число раз, среди операторов нужно предусмотреть оператор, который бы приводил к изменению условия.

Пример. Вычислить бесконечную сумму $S = \sum_{i=1}^{\infty} \frac{1}{i^2}$ с точностью

до члена меньшего ε .

$$S = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots + \frac{1}{n^2} + \dots$$

Для ввода значения ε на рабочем листе установить поля ввода TextBox, значение суммы записать в ячейку B2 (рис. 3.12).

Особенность этой задачи: с увеличением номера i , значение слагаемого соответствующего этому номеру уменьшается. Для любого заданного бесконечно малого ε всегда можно найти такой номер i , для которого $\frac{1}{i^2} < \varepsilon$. Дойдя до такого слагаемого, вычисления прерывают и говорят, что сумма вычислена с точностью ε .

Код модуля рабочего листа Лист1

```
Private Sub CommandButton1_Click()
Dim i As Integer, a, s, eps As Single
eps = Val(TextBox1.Text)
a = 1 : s = 1 : i = 2
Do Until a < eps
a = 1 / i ^ 2
s = s + a
i = i + 1
Loop
Range("B2").Value = Round(s, 3)
End Sub
```

	A	B	C	D	E
1					
2	s	1,558	eps		<input type="text" value="0.01"/>
3					
4					
5				<input type="button" value="Вычислить"/>	
6					

Рис. 3.12.

Задания для самостоятельного выполнения.

1. Вычислить сумму

$$s = \frac{\sin 1}{1} - \frac{\sin 2}{2} + \frac{\sin 3}{3} - \dots + (-1)^{n+1} \frac{\sin n}{n}$$

. Для ввода значения n на рабочем листе установить поля ввода `TextBox`, значение суммы вывести в ячейку рабочего листа.

2. Вычислить сумму $s = \sum_{i=3}^{100} \frac{a(i+1)}{b+i+2}$. Для ввода исходных

данных и вывода результата создать диалоговое окно с

помощью формы. Значения a , b ввести с помощью полей ввода TextBox, значение суммы вывести на надпись.

3. Вычислить сумму $s = \sum_{i=0}^{\infty} \frac{x}{x+i^2}$ с точностью до члена

меньшего ε . Для ввода значений x и ε на рабочем листе разместить два поля ввода TextBox, значение суммы записать в ячейку рабочего листа.

4. Протабулировать функцию $y = \frac{x \sin 3x + x^3 + 5 \cos x}{\sqrt{x^2 + 3}}$ на

отрезке $[a, b]$ с шагом h . Для ввода исходных данных и вывода результата создать диалоговое окно с помощью формы. Значения a , b , h ввести с помощью полей ввода TextBox, таблицу значений аргумента и функции вывести на надпись.

5. Протабулировать функцию $y = \sqrt{x+1} + \frac{1}{x}$ на отрезке $[a, b]$ с

шагом h . Для ввода значений a , b , h на рабочем листе разместить три поля ввода TextBox, для вывода таблицы значений аргумента и функции – список ListBox.

6. Вывести строчные английские буквы в обратном порядке от 'z' к 'a'. Использовать функцию Chr(N), которая возвращает строку из одного символа, соответствующего коду символа с номером N. N – число между 0 и 255. Коды строчных букв английского алфавита от 97 до 122.
7. Дано натуральное число n . Найти сумму квадратов всех его цифр. Для ввода исходных данных и вывода результата создать диалоговое окно с помощью формы. Значение n ввести с помощью поля ввода TextBox, значение суммы вывести на надпись.
8. Дано натуральное число n . Проверить, является ли оно трехзначным, кратным пяти. Для ввода исходных данных и вывода результата создать диалоговое окно с помощью формы. Значение n ввести с помощью поля ввода TextBox, ответ вывести на надпись.

4. Строки

Строка - это последовательность любых символов.

Строки в тексте программы заключаются в двойные кавычки

Строка может быть **пустой**, то есть не содержать ни одного символа. Для обозначения пустой строки кавычки пишут без пробелов между ними: `a = ""`

Объявление строковой переменной:

```
Dim Имя_переменной As string
```

Операции со строками

Для данных типа `String` существует только одна операция – конкатенация `&` (объединение). Например,

```
Dim a As Single
```

```
a = 12.456
```

```
Label1.Caption = "a= " & a
```

Получаем `a = 12.456`

Возможно также использование другого знака – знака плюс (`+`) для операции конкатенации. В нашем примере можно было бы написать:

```
Label1.Caption = "a= " + Str(a) , получили бы тот же результат.
```

Разница между этими выражениями состоит в том, что в первом случае операндами могут быть значения любого типа (они просто будут преобразовываться в строковые данные), а во втором – все операнды должны иметь тип `String`.

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:

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

Пример:

```
Dim a As String
```

```
a = "information"
```


If a Like "informatio[t,i,o,n,h]" Then Label1.Caption = "yes" Else
Label1.Caption = "no"

Получаем «yes».

Выражение а Like "informa[t,i,o,n,h]ion" тоже будет иметь значение True.

Для работы со строками существуют большое количество функций.

Следующие функции обеспечивают преобразование строк:

LCase (<строка>) – преобразует все символы строки к нижнему регистру, например, функция LCase(“СТРОКИ”) возвращает строку “строки”.

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

Следующие функции генерируют строки символов:

Space (<длина>) – создает строку , состоящую из указанного числа пробелов.

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

Функции работы со строками

Функция	Описание
Len(<строка>)	Определяет длину строки
Left(<строка>,<длина>)	Выделяет из аргумента <строка> указанное количество символов слева
Right(<строка>,<длина>)	Выделяет из аргумента <строка> указанное количество символов справа
Mid(<строка>, <старт>,<длина>)	Выделяет из аргумента <строка> подстроку с указанным количеством символов, начиная с позиции <старт>
Mid(<строка>, <старт>)	Выделяется подстрока от позиции <старт> до конца строки
LTrim(<строка>)	Удаляет пробелы в начале строки
RTrim(<строка>)	Удаляет пробелы в конце строки
Trim(<строка>)	Удаляет пробелы в начале и конце строки
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.
InStrRev([<старт>], <строка1>, <строка2> [,<сравнение>])	Производит поиск подстроки в строке, но начинает поиск с конца строки. Возвращает позицию последнего вхождения подстроки. Не обязательный аргумент <сравнение> определяет тип сравнения двух строк.
Replace(<строка>, <строка Поиск>, <строка Замена>)	Позволяет заменить в строке одну подстроку другой. Эта функция ищет все вхождения аргумента <строка Поиск> в аргументе <строка > и заменяет их на <строка Замена>

Пример 1. Дана строка текста, содержащая цифры. Найти сумму цифр. Для ввода текста на рабочем листе установить поле ввода TextBox, значение суммы записать в ячейку B2 (рис.4.1).

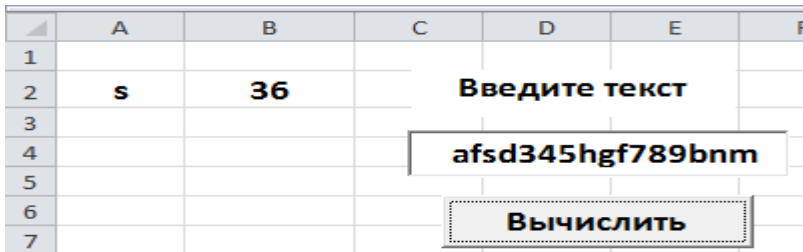


Рис. 4.1.

Код модуля рабочего листа Лист1

```
Private Sub CommandButton1_Click()
Dim s1, s2 As String, i, s As Integer
s1 = TextBox1.Text :s = 0
For i = 1 To Len(s1)
s2 = Mid(s1, i, 1)
If (s2 >= "0" And s2 <= "9") Then s = s + Val(s2)
Next
Range("B2").Value = s
End Sub
```

Пример 2. Дана строка текста. Получить новую строку, заменив каждую группу символов вида “ab” на “*ab*”. Для ввода исходного текста и вывода результата на окне формы установить два поля ввода TextBox (рис. 4.2).

Код модуля формы

```
Private Sub CommandButton1_Click()  
Dim s1, s2 As String  
s1 = TextBox1.Text  
s2 = Replace(s1, "ab", "*ab*")  
TextBox2.Text = s2  
End Sub
```

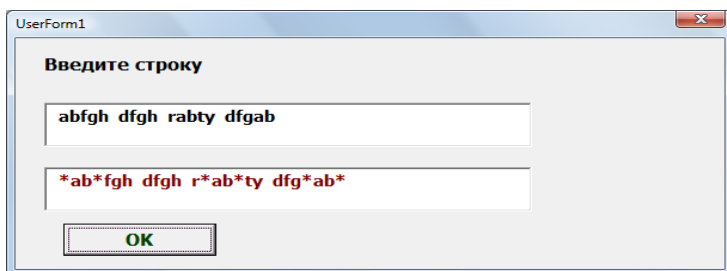


Рис. 4.2. Вид окна программы

Задания для самостоятельного выполнения.

1. В данной строке текста все слова перевернуть. Для ввода исходного текста и вывода результата на окне формы установить два поля ввода TextBox.
2. Дана строка текста. Сформировать новую строку, вставив между встречающимися рядом символами "к" и "м" символ "у". Для ввода исходного текста и вывода результата на рабочем листе установить два поля ввода TextBox.
3. Дана строка текста, в которой есть хотя бы один пробел. Подсчитать количество символов "а" в слове, следующем после первого появления пробела. Для ввода исходного текста на рабочем листе установить поле ввода TextBox, ответ записать в ячейку рабочего листа.
4. Дана строка текста. Найти слова, которые содержат сочетание "из". Для ввода исходного текста и вывода результата на окне формы установить два поля ввода TextBox.

5. Дана строка символов. Встречаются ли в строке пять символов "\$" и три символа "%"? Для ввода исходного текста на окне формы установить поле ввода TextBox, ответ вывести на надпись.
6. Дана строка символов. Заменить все символ "x" на символ "y", а символы "z" на символ "t". Для ввода исходного текста и вывода результата на рабочем листе установить два поля ввода TextBox.

5. Массивы

5.1. Одномерные массивы

Массив — это переменная, в которой хранится одновременно несколько значений одинакового типа, доступ к которым осуществляется по индексу (порядковому номеру). Таким образом, массив представляет собой совокупность однотипных индексированных данных.

Массивы объявляются с помощью оператора Dim:

Dim <имяМассива> (<размер1>,<размер2>,...) As тип данных

где указанные в скобках величины <размер1>,<размер2>,... задают размеры массива – количество индексов и максимально допустимое значение для каждого конкретного индекса. По умолчанию индексирование элементов массива начинается с нуля.

Пример: Dim A(3) As Integer

Создается массив по имени A, состоящий из четырех элементов типа Integer.

Примечание

В качестве стандартного значения нижней границы массива (индекса) может использоваться не только ноль. Чтобы изменить это стандартное значение, нужно воспользоваться оператором Option Base. Например, если в начале модуля разместить оператор Option Base 1, то индексирование элементов массивов по умолчанию будет начинаться не с нуля, а с единицы.

Другим способом изменения базового индекса является использование ключевого слова To при объявлении массива:

Dim A (1 To 3) As Integer

В большинстве программ при создании массива сразу же инициализируют его, присвоив каждому элементу, нулевое значение или пустую строку.

Dim Sotrudnik (5) As String, i As Integer

```

For i=0 to 5
Sotrudnik(i) = ""
Next i

```

Обычно элементы массива содержат значения одного и того же типа. Если же необходимо, чтобы в массиве содержались данные разных типов, при объявлении массива нужно указать тип Object:

```
Dim A (2) As Object
```

Элементы такого массива могут содержать значения разных типов:

```
A (0) = "Никитин"
```

```
A (1) = 56
```

```
A (2) = 3.1415
```

5.2. Двумерные массивы

При создании двумерных массивов нужно указать количество строк и столбцов.

```
Dim B (1 To 2, 1 To 2) As Single
```

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

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

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

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

Создается двумерная таблица:

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

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

Функция Array (список значений) создает массив типа Variant. Аргумент в скобках представляет разделенный запятыми список значений, присваиваемых элементам массива.

В VBA имеются две функции, которые возвращают нижнее и верхнее граничные значения индексов массива:

```
Lbound (имя массива [,размерность])
```

```
Ubound (имя массива [,размерность])
```

где имя массива – 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.

Процедура Erase используется для очистки массива. Позволяет очищать все элементы массива, в основном переустанавливая массив в

то же самое состояние, какое он имел, когда VBA создавал его в оперативной памяти.

Вызов процедуры Erase имеет следующий вид:

Erase array1 [,array2, ...]

Здесь array1 ,array2 – любые допустимые имена массивов VBA.

Следующие три функции позволяют работать с массивами строк.

Split (<строка> [,<разделитель>]) - преобразует строку в массив подстроки. 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.

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

Filter (<массив строк>,<строка поиск> [,<параметр>] [,<сравнение>]) просматривает массив строковых значений и ищет в нем все подстроки, совпадающие с заданной строкой. Эта функция имеет четыре аргумента:

строка поиск – искомая строка;

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

В VBA при обработке объектов, составляющих массив или семейство однопольных объектов, часто используется разновидность цикла For ... Next. В этой разновидности счетчик отсутствует, а тело цикла выполняется для каждого элемента массива или семейства объектов. Синтаксис такого цикла следующий:

For Each <элемент> **in** <совокупность>

<блокОператоров>

Next [<элемент>]

<элемент> - это переменная, используемая для ссылки на элементы семейства объектов;

<совокупность> - имя массива или семейства.

Пример 1. Дан массив слов. Найти все слова имеющие сочетание «про». Строку поиска и параметр ввести с помощью TextBox, результат вывести на надпись (рис. 5.1).

Код модуля формы

```
Private Sub CommandButton1_Click()  
Dim a, b As Variant, k As Boolean  
Dim s1, s As String  
' создается массив a  
a = Array("проталина", "надстройка", "проба", "премьера", "просмотр")  
s = TextBox1.Text ' подстрока поиска  
k = True ' параметр  
If TextBox2.Text = "true" Then  
k = True  
Else  
If TextBox2.Text = "false" Then k = False Else MsgBox "Неверный  
параметр", 48  
End If  
' создается массив в из строк, содержащих подстроку поиска  
' или не содержащих в зависимости от значения параметра  
b = Filter(a, s, k)  
' массив в преобразуется в строку s1  
s1 = Join(b)  
' вывод результата  
Label1.Caption = s1  
End Sub
```

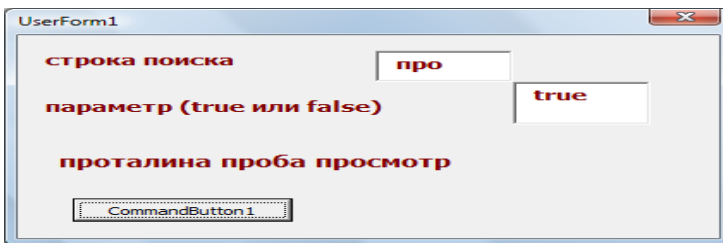


Рис. 5.1. Вид окна программы

Пример 2. Дан числовой массив $A(n)$. Найти среднее арифметическое элементов массива. Элементы массива ввести с помощью `TextBox`. При вводе элементы массива разделить одним пробелом. Результат вывести на надпись (рис. 5.2).

Код модуля формы

```
Private Sub CommandButton1_Click()
```

```

Dim a As Variant, s, i, n As Integer
If TextBox1.Text <> "" Then
    ' Преобразование строки в массив
    a = Split(TextBox1.Text)
    ' Определение верхней границы массива
    n = UBound(a)
    s = 0
    ' вычисление суммы элементов массива
    For Each b in a
        s=s+b
    Next
    Label1.Caption = "Среднее арифметическое: " & Format(s / (n + 1),
"###0.00")
    Else
        MsgBox "Массив не введен", 48
    End If
End Sub

```

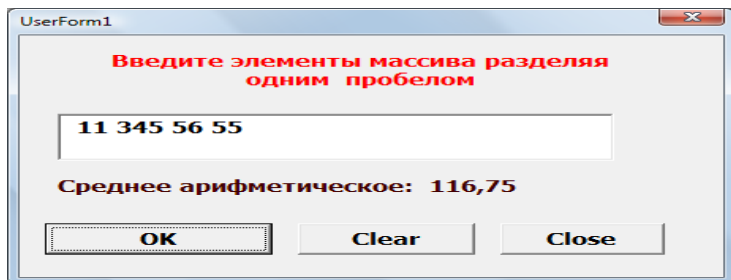


Рис. 5.2. Вид окна программы

Пример 3. Дан числовой массив $A(n)$. Максимальный и минимальный элементы массива переставить местами. Для ввода массива на рабочем листе установить поле ввода `TextBox`, для вывода результата – список `Listbox`, для управления работой программы – две командные кнопки (рис. 5.3).

Код модуля рабочего листа Лист1

```

Private Sub CommandButton1_Click()
Dim i, n, max, min, k, m As Integer
Dim a As Variant
If TextBox1.Text = "" Then

```



```

MsgBox "Массив не введен", 48
Exit Sub
End If
' Преобразование строки в массив
a = Split(TextBox1.Text)
' Определение верхней границы массива
n = UBound(a)
' Поиск максимального и минимального
' элементов массива
max = Val(a(0)): min = Val(a(0))
k = 0: m = 0
For i = 1 To n
If Val(a(i)) > max Then
max = Val(a(i)): k = i
End If
If Val(a(i)) < min Then
min = Val(a(i)): m = i
End If
Next
a(k) = Str(min): a(m) = Str(max)
For i = 0 To n
ListBox1.AddItem (a(i))
Next
End Sub

Private Sub CommandButton2_Click()
TextBox1.Text = ""
ListBox1.Clear
End Sub

```

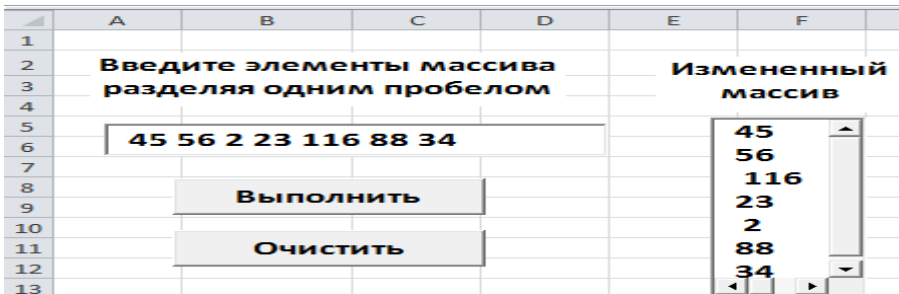


Рис. 5.3.

Пример 4. На листе Excel дан массив целых чисел, который начинается с ячейки B2 (рис. 5.4). Массив отобразить на списке ListBox. Число строк и столбцов массива ввести с помощью элементов управления TextBox. Над элементами выбранной строки массива произвести одно из следующих действий: суммирование, вычисление произведения или среднего арифметического в зависимости от выбора действия. Действие выбрать с помощью одного из трех переключателей OptionButton.

	A	B	C	D	E	F	G
1							
2		4	23	15	7	24	
3		5	12	37	8	19	
4		6	56	48	10	27	
5		31	43	27	35	5	
6		61	16	1	3	81	
7							

Рис. 5.4.

Вид диалогового окна программы приведен на рис. 5.5. После ввода числа строк и столбцов, чтобы массив отобразить на списке выбора ListBox нажать кнопку «Отобразить». Чтобы выполнить выбранное действие над элементами выбранной строки и вывода результата нажать кнопку «Вычислить».

Рис. 5.5. Вид окна программы

Код модуля формы

```
Dim n, m, i, j, sum As Integer
Dim sr As Double, p As Long
Dim s1 As String
Private Sub CommandButton1_Click()
n = Val(TextBox1.Text)
m = Val(TextBox2.Text)
```

```

' Если введены некорректные данные,
' осуществляется выход из процедуры
If (n <= 0) Or (m <= 0) Then
MsgBox "Введите положительные числа", 48
Exit Sub
End If
'В s1 адрес диапазона, откуда считывается массив
s1 = "B2:" + Chr(66 + m - 1) + Chr(49 + n)
'Коды цифр - 48 – 57
'Коды прописных английских букв от А до Z - 65 – 90
Range(s1).Select ' Выбор и выделение диапазона
With ListBox1
.ColumnCount = m 'Устанавливается число столбцов списка
.RowSource = (s1) ' массив отображается на списке
End With
End Sub

Private Sub CommandButton2_Click()
Dim s2 As String
If OptionButton1.Value = True Then
sum = 0
With ListBox1
For i = 0 To n - 1
If .Selected(i) = True Then
For j = 0 To m - 1
sum = sum + .List(i, j)
Next j
End If
Next i
End With
Label3.Caption = Label3.Caption + " " + Str(sum)
End If

If OptionButton2.Value = True Then
p = 1
With ListBox1
For i = 0 To n - 1
If .Selected(i) = True Then

```

```

        For j = 0 To m - 1
            p = p * .List(i, j)
        Next j
    End If
Next i
End With
Label3.Caption = Label3.Caption + " " + Str(p)
End If

If OptionButton3.Value = True Then
sum = 0
With ListBox1
    For i = 0 To n - 1
        If .Selected(i) = True Then
            For j = 0 To m - 1
                sum = sum + .List(i, j)
            Next j
        End If
    Next i
End With
sr = sum / m
Label3.Caption = Label3.Caption + " " + Str(sr)
End If
End Sub

```

Задания для самостоятельного выполнения:

1. Дан числовой массив $A(n)$. Найти произведение и среднее геометрическое положительных элементов массива. Элементы массива ввести с помощью `TextBox`. Если k – число положительных элементов массива, p – их произведение, то среднее геометрическое вычисляется по формуле $k\sqrt[p]{p}$ (корень k -ой степени из p). Результат вывести на надпись.
2. Дан массив слов. Есть ли в этом массиве слова-палиндромы (т. е. слова, одинаково читающиеся слева направо и справа налево)? Если есть, то вывести их на экран. Для ввода исходного массива слов на рабочем листе установить поле ввода `TextBox`, для вывода результата – список `ListBox`.

3. Дан массив $A(n)$. Подсчитать, сколько в массиве элементов, больше первого. Для ввода массива на окне формы установить поле ввода `TextBox`. Результат вывести на надпись.
4. Дан целочисленный массив $A(n)$. Вычислить произведение элементов массива, кратных семи. Для ввода массива на рабочем листе установить поле ввода `TextBox`, результат записать в ячейку рабочего листа.
5. Дан массив слов. Найти самое длинное слово. Для ввода исходного массива на рабочем листе установить поле ввода `TextBox`, результат записать в рабочую ячейку.
6. Дан массив слов. Найти слова начинающиеся буквой *a* и оканчивающиеся буквой *y*. Для ввода исходного массива на рабочем листе установить поле ввода `TextBox`, для вывода результата – список `ListBox`.
7. На листе Excel дан массив целых чисел, который начинается с ячейки A1. Массив отобразить на списке выбора `ListBox`. Число строк и столбцов массива ввести с помощью элементов управления `TextBox1` и `TextBox2`. Над элементами выбранной строки массива произвести одно из следующих действий: суммирование четных элементов, вычисление произведения отрицательных элементов или среднего геометрического положительных элементов в зависимости от выбора действия. Действие выбрать с помощью одного из трех переключателей `OptionButton`.

5.3. Динамические массивы

Если заранее неизвестно, сколько будет введено данных в массив, или объем данных собираемых для массива значительно меняется, то в подобных ситуациях можно создать динамический массив. Динамические массивы создаются с помощью оператора `Dim`, `Private`, `Public`, причем список размерностей опускается, затем их размер устанавливается с помощью оператора `ReDim` во время выполнения процедуры.

Оператор `ReDim` имеет следующий синтаксис:

```
ReDim [Preserve] ИмяПеременной (индексы) [As тип] _
[,ИмяПеременной (индексы) [As тип] ]
```

Здесь необязательное ключевое слово *Preserve* приводит к тому, что VBA сохраняет данные в имеющемся массиве;

ИмяПеременной - имя существующего массива;

индексы – измерения массива;

тип – любой тип VBA или определенный пользователем тип.

Примеры:

- 1) Dim Month() As String - объявляет динамический массив Month;
- 2) Redim Month(1 To 30) – изменяет размер массива до 30 элементов;
- 3) Redim Preserve Month(1 To 31) – изменяет размер массива до 31 элемента, сохраняя содержимое;
- 4) Dim Table() As Integer – объявляет динамический массив;
- 5) ReDim Table(3, 15) – делает массив двумерным;
- 6) ReDim Table(4, 20) – изменяет размер двумерного массива;
- 7) ReDim Preserve Table(4, 24) – только изменяет последний размер массива;
- 8) Dim Mas As Variant – объявляет переменную типа Variant;
- 9) ReDim Mas(20) As Integer – создает массив из 20 целых чисел в Variant.

Пример 1. Сформировать двумерный массив $A(n,m)$ из случайных чисел. Случайные числа умножить на 100 и округлить до ближайшего целого. Число столбцов массива увеличить на единицу. Вычислить сумму элементов каждой строки и записать в добавленный столбец. Массив отобразить на элементе управления ListBox (рис. 5.6).

Код модуля формы

```
' Объявляем динамический массив
Dim a() As Single
Dim m, n, i, j As Integer, s As Single
Private Sub CommandButton1_Click()
n = Val(TextBox1.Text)
m = Val(TextBox2.Text)
' Если введены некорректные данные,
' осуществляется выход из процедуры
If (n<= 0) Or (m<=0) Then
MsgBox "Введите положительные числа", 48
Exit Sub
End If
' Устанавливаем размерности массива
ReDim a(1 To n, 1 To m)
```

```

' Формируем массив из случайных чисел
For i = 1 To n
  For j = 1 To m
    a(i, j) = Int(Rnd(20) * 100)
  Next
Next
' Отображаем массив на ListBox
  With ListBox1
    .ColumnCount = m
    .List = a
  End With
End Sub

Private Sub CommandButton2_Click()
' Очищаем ListBox
ListBox1.Clear
' Число столбцов массива увеличиваем на 1,
' сохраняя содержимое
  ReDim Preserve a(1 To n, 1 To m + 1)
' Вычисляем сумму элементов каждой строки и
' записываем в добавленный столбец
  For i = 1 To n
    s = 0
    For j = 1 To m
      s = s + a(i, j)
    Next
    a(i, m + 1) = s
  Next
' Измененный массив отображаем на ListBox
  With ListBox1
    .ColumnCount = m + 1
    .List = a
  End With
End Sub

Private Sub CommandButton3_Click()
ListBox1.Clear
TextBox1.Text = ""

```

```

TextBox2.Text = ""
End Sub

Private Sub CommandButton4_Click()
UserForm1.Hide
End Sub

```

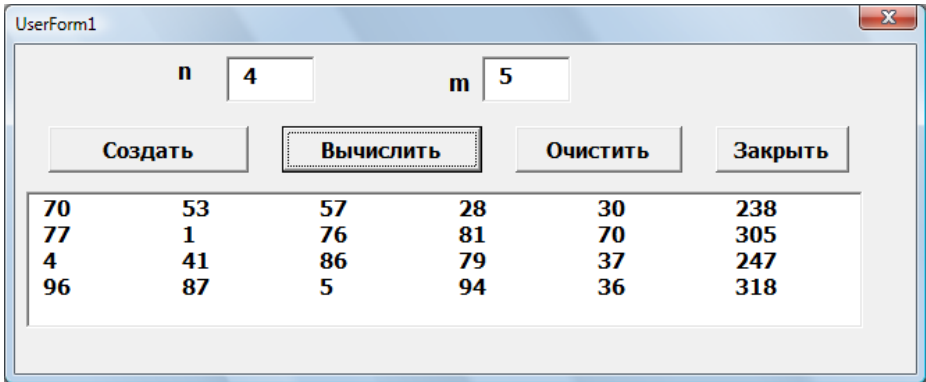


Рис. 5.6. Вид окна программы

Задания для выполнения

1. Дан двумерный массив A размера $n \times n$. Найти произведение ненулевых элементов массива, находящихся выше главной диагонали. Исходную матрицу вывести на элемент управления `ListBox`, произведение – на надпись.
2. Дан двумерный массив A размера $n \times n$. Элементы каждой строки разделить на диагональный элемент. При формировании измененной матрицы элементы округлить до двух знаков после запятой. Результат вывести на элемент управления `ListBox`.
3. Сформировать двумерный массив $x(n,m)$ по правилу $x(i,j) = i^2 + j^2$. Число столбцов массива увеличить на единицу. Вычислить среднее арифметическое элементов каждой строки и записать в добавленный столбец. Результат вывести на элемент управления `ListBox`.
4. Дан двумерный массив A размера $n \times n$. Все элементы ниже главной диагонали увеличить на 3. Результат вывести на элемент управления `ListBox`.
5. Сформировать двумерный массив $y(n,m)$ по правилу $y(i,j) = (i+1) \cdot (j+2)$. Число столбцов массива увеличить на единицу. Вычислить сумму элементов кратных трем каждой строки и

записать в добавленный столбец. Результат вывести на элемент управления ListBox.

6. Дан двумерный массив A размера $n*m$. Подсчитать количество и сумму всех элементов массива, кратных 5. Исходную матрицу вывести на элемент управления ListBox, количество и сумму элементов кратных трем – на надпись.

6. Процедуры и функции

Процедуры и функции представляют собой относительно самостоятельные фрагменты программы, оформленные особым образом и снабженные именем. Упоминание этого имени в тексте программы называется *вызовом* процедуры (функции). В отличие от процедуры имя функции выступает также в качестве переменной и используется для возвращения значения в точку вызова функции.

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, `CommandButton1_Click` – процедура обработки события нажатия кнопки `CommandButton1`.

Общие процедуры не связаны с конкретным объектом. Они выполняются только тогда, когда вызываются другими процедурами. Обычно эти процедуры выполняют какие-то действия и могут вызываться разными процедурами обработки событий.

Процедуры, как и переменные, должны быть объявлены до их вызова. Объявления общих процедур помещаются в разделе `General` (общая часть) модуля.

Процедура заключается между операторами `Sub` и `End Sub`, функция – `Function` и `End Function`.

6.1. Описание и вызов процедур и функций

Описание процедуры:

```
[Private | Public] Sub <имя процедуры> [(<аргументы>)]  
<инструкции>  
[Exit Sub]  
<инструкции>  
End Sub
```

<аргументы> – список параметров, значения которых передаются в процедуру или возвращаются из процедуры при ее вызове. Разделителем в списке параметров является запятая.

Список параметров процедуры [(**<аргументы>**)] имеет следующий синтаксис:

[Optional] [ByVal|ByRef] [ParamArray] ИмяПеременной [As Тип]

Optional – ключевое слово, указывающее на то, что параметр не является обязательным. При использовании этого элемента все последующие параметры, которые содержит список <аргументы>, также должны быть необязательными, и их надо описать с помощью ключевого слова Optional. Все параметры, описанные как Optional, должны иметь тип Variant. Не допускается использование ключевого слова Optional для любого из параметров, если указано ключевое слово ParamArray.

ByVal - ключевое слово, указывающее на то, что этот параметр передается по значению.

ByRef - ключевое слово, указывающее на то, что этот параметр передается по ссылке. Описание ByRef используется по умолчанию.

ParamArray - ключевое слово, которое используется только в качестве последнего элемента в списке <аргументы> для указания, что конечным параметром является описанный как Optional массив значений типа Variant. Ключевое слово ParamArray позволяет задавать произвольное количество параметров. Оно может быть использовано с ключевыми словами ByVal, ByRef или Optional.

Тип – тип значений параметра, переданного процедуре. Допустимы значения: Byte, Boolean, Integer, Long, Currency, Single, Double, Data, String, Object, Variant. Если отсутствует ключевое слово Optional, может быть также указан определяемый пользователем тип или объектный тип.

Описание функции:

[Private | Public] Function < имя функции > [(**<аргументы>**)] _ [As <тип результата>]

<инструкции>

< имя функции > = < возвращаемое значение >

[Exit Function]

<инструкции>

< имя функции > = < возвращаемое значение >

End Function

В теле функции обязательно присутствует оператор присвоения имени функции возвращаемого функцией результата. В заголовке описывается тип возвращаемого функцией результата. Если этот тип не указан тип возвращаемого результата будет Variant.

Вызов процедур и функций

Чтобы использовать написанную процедуру или функцию, ее нужно вызвать. Процедура может вызываться двумя способами:

< 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. Список фактических параметров должен полностью соответствовать списку формальных параметров. Все фактические параметры должны быть перечислены в том порядке, в каком они присутствуют в описании процедуры.

Вызов функции имеет следующий вид:

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

Список фактических параметров при вызове функции обязательно должен заключаться в скобки.

Возможны два разных способа передачи переменных процедуре или функции: по ссылке и по значению. Если переменная передается *по ссылке*, то это означает, что процедуре или функции будет передан адрес этой переменной в памяти. При этом происходит отождествление формального аргумента процедуры и переданного ей фактического параметра. Поэтому вызываемая процедура может изменить значение фактического параметра: если будет изменен формальный аргумент процедуры, то это скажется на значении переданного ей при вызове фактического параметра. Если же фактический параметр передается *по значению*, то формальный аргумент вызываемой процедуры или функции получает только значение фактического параметра, но не саму переменную, используемую в качестве этого параметра. Тем самым все изменения значения формального аргумента не скажутся на значении переменной, являющейся фактическим параметром.

Способ передачи параметров процедуре или функции указывается при описании ее аргументов: имени аргумента может предшествовать явный описатель способа передачи. Описатель `ByRef` определяет передачу по ссылке, а `ByVal` – по значению. По умолчанию подразумевается передача по ссылке. Например, `Primer1(x, ByVal y, ByRef z)`. Первый параметр передается по ссылке (по умолчанию), второй – по значению, третий – по ссылке.

Значения, хранящиеся в ячейках электронной таблицы и передаваемые в функцию через список параметров, можно изменять в функции, однако, в электронной таблице эти значения останутся неизменными. С помощью возвращаемого функцией значения можно изменять значения ячеек, однако хранящиеся в ячейках значения, передаваемые в функцию через список параметров, не изменятся после вызова функции, т.е. эти величины передаются в функцию только по значению.

Ключевые слова `Public` и `Private`

Эти ключевые слова определяют область видимости процедуры. Ключевое слово *Private* делает процедуру доступной только на уровне модуля – это означает, что работать с ней могут лишь объекты, находящиеся в той же форме или модуле. Процедуры и функции, объявленные ключевым словом *Public*, доступны на уровне всего проекта, т.е. имеют *глобальную область видимости*.

Все процедуры обработки событий начинаются с ключевого слова *Private*. Оно означает, что процедура не может быть вызвана за пределами модуля. Ее область видимости ограничивается модулем, она является локальной, или *закрытой процедурой*. По умолчанию все процедуры обработки событий объявляются с ключевым словом *Private*.

Общие процедуры, объявленные ключевым словом *Public*, доступны на уровне приложения, такие процедуры называются открытыми или глобальными. Если ключевое слово *Private* или *Public* не указано, по умолчанию процедура считается открытой.

6.2. Создание пользовательских функций

Для создания пользовательской функции в проект нужно добавить стандартный модуль. Добавление стандартного модуля осуществляется командой **Insert, Module**. Рассмотрим следующий пример. Пусть нам нужно вычислить значение следующего выражения

$C = \frac{n!}{m!(n-m)!}$. Для решения данной задачи создадим

пользовательскую функцию вычисления значения факториала по

следующей формуле $F = \begin{cases} 0, & \text{если } n < 0 \\ 1, & \text{если } n = 0 \\ n!, & \text{если } n > 0 \end{cases}$

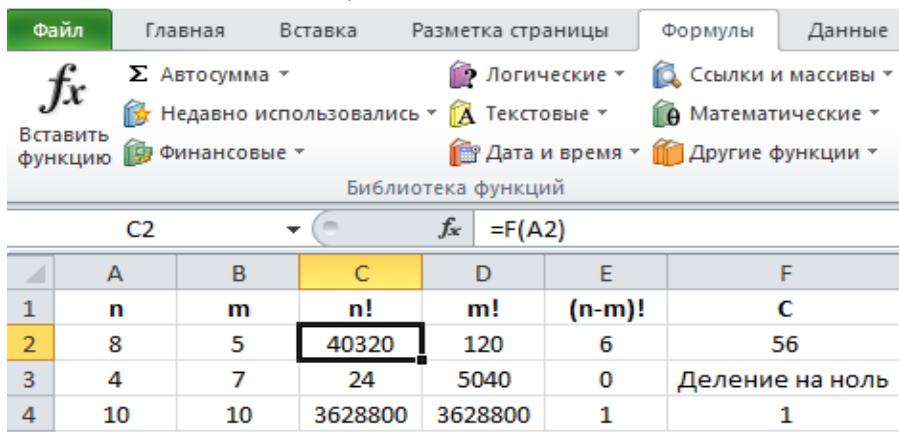


Рис. 6.1.

Для реализации данной задачи выполним следующие действия.

1. В окне редактора VBA добавим лист стандартного модуля, выполнив команду **Insert, Module**.
2. В окне созданного модуля наберем следующий код:

```
Function F(n As Integer)
```

```
Dim f1 As Long
```

```
If n < 0 Then f1 = 0
```

```
If n = 0 Then f1 = 1
```

```
If n > 0 Then
```

```
f1 = 1
```

```
For i = 1 To n
```

```
f1 = f1 * i
```

```
Next
```

```
End If
```

```
F = f1
```

```
End Function
```

По умолчанию созданная пользовательская функция попадает в

раздел **Определенные пользователем** списка **Категория** окна **Мастер функций**.

3. Перейдем в окно рабочего листа **Лист1**, в ячейках A2 и B2 введем значения n и m (рис.6.1).
4. Перейдем в ячейку C2, где будет вычисляться $n!$.
5. Перейдем на вкладку **Формулы**, в группе **Библиотека функций** нажмем кнопку **Вставить функцию**.
6. Откроется окно **Мастер функций**, из списка **Категория** выберем **Определенные пользователем**, выберем функцию F и нажмем кнопку ОК.

В ячейки D2 и E2 вставим функции для вычисления $m!$ и $(n-m)!$, в ячейку F2 вставим логическую функцию **Если**
 $=\text{ЕСЛИ}(\text{И}(\text{E2}>0;\text{D2}>0);\text{C2}/(\text{D2}*\text{E2});\text{"Деление на ноль"})$

В пользовательских функциях в качестве значений параметров можно использовать ссылки на диапазоны ячеек. Рассмотрим следующий пример. Пусть нам нужно суммировать значения для указанного диапазона ячеек. Для решения данной задачи создадим пользовательскую функцию Sumd. В качестве параметра данной функции передадим ссылку на диапазон ячеек. Ячейку B5 отведем под сумму. В ячейку B5 вставим вызов функции Sumd(A1:D4) (рис. 6.2).

		B5			
		fx =sumd(A1:D4)			
	A	B	C	D	E
1	6	87	-56	76	
2	-34	-23	8	12	
3	65	-94	-87	-36	
4	44	41	27	81	
5	Сумма	117			

Рис. 6.2.

Код Module1

Function sumd(rng As Range)

Dim c As Range

Dim s As Double

s = 0

For Each c in rng.Cells

s = s + c.Value

```

Next
sumd = s
End Function

```

Пример 1. Даны два треугольника с двумя сторонами и углом между ними. Определить площадь какого из треугольников больше. Для ввода исходных данных и вывода результата создать диалоговое окно (рис. 6.3).

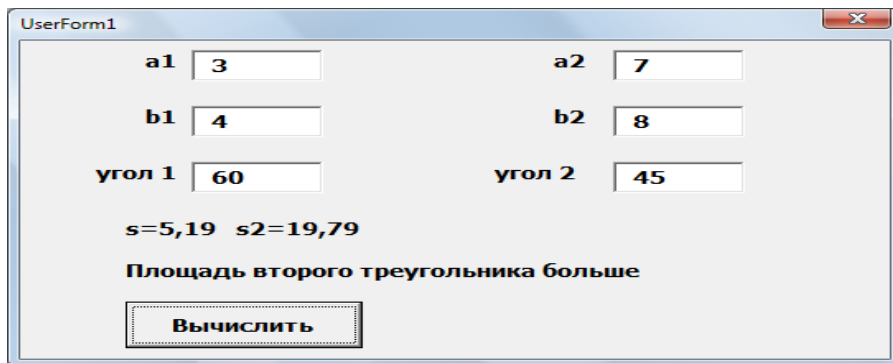


Рис. 6.3. Вид окна программы

Зная две стороны и угол между ними по теореме косинусов можно найти третью сторону треугольника $c = \sqrt{a^2 + b^2 - 2ab \cos \alpha}$, где a, b, c – стороны треугольника, α - угол между сторонами a и b. Для определения третьей стороны треугольника воспользуемся функцией st3.

Зная три стороны площадь треугольника можно вычислить по формуле Герона $s = \sqrt{p(p-a)(p-b)(p-c)}$, где a, b, c – стороны треугольника, p – полупериметр $p = \frac{a+b+c}{2}$. Для вычисления площади треугольника воспользуемся процедурой Geron.

Код модуля формы

```

Const pi As Double = 3.14
' Функция определения третьей стороны треугольника
Function st3(a, b, x As Variant) As Single
st3 = Sqr(a ^ 2 + b ^ 2 - 2 * a * b * Cos(pi * x / 180))
End Function

```

```

' Процедура вычисления площади треугольника
Sub Geron(a, b, c, s As Variant)
Dim p As Single
p = (a + b + c) / 2
s = Sqr(p * (p - a) * (p - b) * (p - c))
End Sub

Private Sub CommandButton1_Click()
Dim s As String
Dim a1, b1, a2, b2, s1, s2, c1, c2 As Single, x1, x2 As Integer
s = ""
' Ввод исходных данных
a1 = Val(TextBox1.Text)
b1 = Val(TextBox2.Text)
x1 = Val(TextBox3.Text)
a2 = Val(TextBox4.Text)
b2 = Val(TextBox5.Text)
x2 = Val(TextBox6.Text)
c1 = st3(a1, b1, x1)
c2 = st3(a2, b2, x2)
Geron a1, b1, c1, s1
Geron a2, b2, c2, s2
' вывод результата
s = s + "s1=" + Format(s1, "###0.00") + " s2=" + Format(s2, "###0.00") +
Chr(13) + Chr(13)
If s1 > s2 Then s = s + "Площадь первого треугольника больше" Else If s1
= s2 Then s = s + "Площади равны " Else s = s + " Площадь второго
треугольника больше "
Label7.Caption = s
End Sub

```

Пример 2. Даны массивы $x(n)$, $y(m)$, $z(k)$ из случайных чисел. Вычислить среднее арифметическое их максимальных элементов. Для ввода количества элементов массивов на рабочем листе установить три поля ввода TextBox, для отображения массивов – три списка ListBox, среднее арифметическое максимальных элементов вывести на надпись (рис. 6.4).

В VBA допустимо использование массива в качестве параметра процедуры. В этом случае массив, используемый в качестве

параметра, при описании процедуры должен быть объявлен как динамический. Параметром процедуры может быть и ссылка на объект, в том числе и ссылка на элемент управления.

Для решения данной задачи создадим две процедуры:

1. Формирование массива из случайных чисел и отображение его на ListBox. Аргументы процедуры: массив, количество элементов массива и ссылка на элемент управления ListBox.
2. Поиск максимального элемента массива. Аргументы процедуры: массив, максимальный элемент массива.

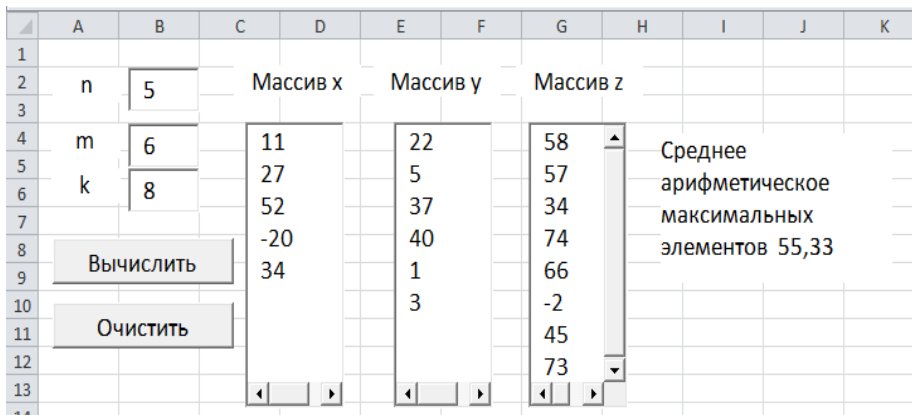


Рис. 6.4.

Код модуля рабочего листа Лист1

' Процедура формирования массива из случайных чисел
' и отображения его на ListBox

```
Sub form(a() As Single, l As Variant, obj1 As Object)
```

```
Dim i As Integer
```

```
For i = 1 To l
```

```
a(i) = Round(Rnd(1) * 100) - 25
```

```
obj1.AddItem (Str(a(i)))
```

```
Next
```

```
End Sub
```

' Процедура поиска максимального элемента массива

```
Sub maxim(a() As Single, max As Variant)
```

```
max = a(1)
```

```
For Each c In a
```

```

If c > max Then max = c
Next
End Sub

Private Sub CommandButton1_Click()
Dim x() As Single, y() As Single, z() As Single
Dim mx, my, mz, crmax As Single
Dim n, m, k As Integer
n = (TextBox1.Text)
m = (TextBox2.Text)
k = (TextBox3.Text)
' Если введены некорректные данные,
' осуществляется выход из процедуры
If (n<= 0) Or (m<=0) Or (k<=0) Then
MsgBox "Введите положительные числа", 48
Exit Sub
End If
' Установление размерности массивов
ReDim x(1 To n) As Single
ReDim y(1 To m) As Single
ReDim z(1 To k) As Single
' Формирование массивов и вывод их на ListBox
Call form(x, n, ListBox1)
Call form(y, m, ListBox2)
Call form(z, k, ListBox3)
' Поиск максимальных элементов массивов
Call maxim(x, mx)
Call maxim(y, my)
Call maxim(z, mz)
' Определение среднего арифметического максимальных
элементов
crmax = (mx + my + mz) / 3
Label7.Caption = "Среднее арифметическое максимальных
элементов " + Format(crmax, "###00.00")
End Sub

Private Sub CommandButton2_Click()
TextBox1.Text = ""

```

```

TextBox2.Text = ""
TextBox3.Text = ""
ListBox1.Clear
ListBox2.Clear
ListBox3.Clear
Label7.Caption = ""
End Sub

```

Пример 4. Вычислить значение выражения $z = x_1 + x_2 + x_3$, где

$$x_1 = \frac{\sum_{i=1}^{n1} a_i}{n1!}, \quad x_2 = \frac{\sum_{i=1}^{n2} b_i}{n2!}, \quad x_3 = \frac{\sum_{i=1}^{n3} c_i}{n3!}$$

a_i, b_i, c_i - элементы массивов

a, b, c ; $n1, n2, n3$ - количество элементов массивов a, b, c соответственно. Для вычисления суммы элементов массива и значения факториала создать пользовательские функции. На рабочем листе установить две командные кнопки «Вычислить» и «Очистить», для ввода элементов массивов - три поля ввода TextBox (рис. 6.5).

	A	B	C	D	E	F	G
1	x1	0,027		Введите массивы			
2	x2	1,483					
3	x3	0,189	a	3 45 6 8 21 45 8			
4	z	1,699	b	4 1 56 42 75			
5							
6			c	5 33 67 9 7 15			
7							
8				Вычислить			
9							
10				Очистить			
11							

Рис. 6.5.

Код Module1

```

' Пользовательские функции
Function Summa(d)
Dim s As Single
s = 0
For Each b In d
s = s + b
Next

```

```
Summa = s  
End Function
```

```
Function Fact(n)  
Dim f As Long, i As Integer  
f = 1  
For i = 1 To n  
f = f * i  
Next  
Fact = f  
End Function
```

Код модуля рабочего листа Лист1

```
Private Sub CommandButton1_Click()  
Dim a, b, c As Variant  
Dim z, x1, x2, x3 As Double  
Dim n1, n2, n3 As Integer  
a = Split(TextBox1.Text)  
b = Split(TextBox2.Text)  
c = Split(TextBox3.Text)  
If (TextBox1.Text = "") Or (TextBox2.Text = "") Or (TextBox3.Text = "") Then  
MsgBox "Массив не введен", 48  
Exit Sub  
End If  
' Определение количества элементов массивов  
n1 = UBound(a)+1  
n2 = UBound(b)+1  
n3 = UBound(c)+1  
x1 = Summa(a) / Fact(n1)  
x2 = Summa(b) / Fact(n2)  
x3 = Summa(c) / Fact(n3)  
' Запись значений x1, x2, x3 в ячейки B1, B2, B3  
Range("B1").Value = Round(x1, 3)  
Range("B2").Value = Round(x2, 3)  
Range("B3").Value = Round(x3, 3)  
z = x1 + x2 + x3  
' Запись значения z в ячейку B4  
Range("B4").Value = Round(z, 3)
```

```
End Sub
Private Sub CommandButton2_Click()
    TextBox1.Text = ""
    TextBox2.Text = ""
    TextBox3.Text = ""
    Range("B1:B4").Clear
End Sub
```

6.3. Создание процедур обработки ошибочных ситуаций

При составлении приложений важно предусмотреть, чтобы программа анализировала возможные ошибки, возникающие при ее выполнении, и информировала об этом. При этом возможно два подхода:

1. Предотвращение ошибок: программно анализировать вводимые или вычисляемые данные и в случае, если они могут приводить к ошибке, обеспечить, чтобы программа информировала о необходимости корректного задания данных.
2. Обработка ошибок: в случае возникновения ошибки, перехватить ее, обработать и программно откликнуться на возникшую ошибку.

Рассмотрим пример создания приложения, в котором предотвращается возникновение ошибочных ситуаций:

Дан числовой массив $a(n)$. Найти среднее арифметическое отрицательных элементов массива. Для ввода массива и вывода результата создать диалоговое окно (рис. 6.6).

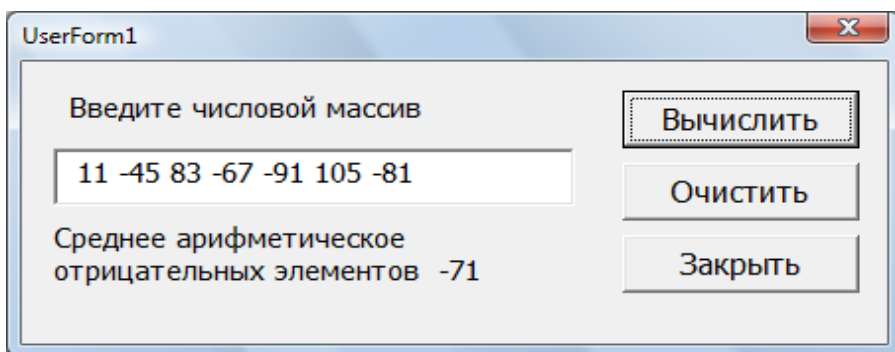


Рис. 6.6.

Для предотвращения аварийного завершения программы анализируются вводимые данные. При вводе некорректных данных

т.е. данных, которые не могут быть преобразованы в число, выдается сообщение об ошибке и осуществляется выход из процедуры.

Если в массиве не окажется отрицательных элементов, возникает ситуация деления нуля на ноль, произойдет аварийная остановка выполнения программы с отображением в диалоговом окне **Microsoft Visual Basic** сообщения: **Overflow** (рис. 6.7). Для избежания подобной ошибки проверяется число отрицательных элементов массива. Если в массиве нет отрицательных элементов об этом выводится сообщение и осуществляется выход из процедуры.

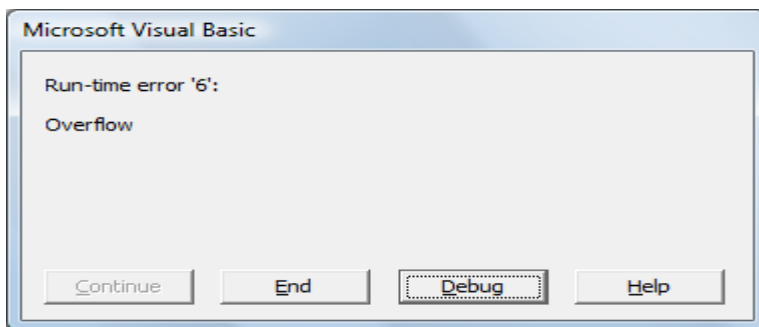


Рис. 6.7.

Код модуля формы

```
Private Sub CommandButton1_Click()  
Dim a As Variant  
Dim i, k As Integer  
Dim s As Single  
If TextBox1.Text = "" Then  
    MsgBox "Массив не введен", 48, "Среднее арифметическое  
    отрицательных элементов"  
    Exit Sub  
End If  
  
a = Split(TextBox1.Text)  
' Проверка, являются ли элементы массива числами  
For Each b In a  
    If IsNumeric(b) = False Then  
        MsgBox "Ошибка в записи числа "+b, 48, _  
        "Среднее арифметическое отрицательных элементов"  
        TextBox1.SetFocus
```

```

Exit Sub
End If
Next
s = 0: k = 0
For Each b In a
    If Val(b) < 0 Then
        s = s + Val(b) : k = k + 1
    End If
Next
If k = 0 Then
    MsgBox "В массиве нет отрицательных элементов ", 48, _ "Среднее
арифметическое отрицательных элементов"
Exit Sub
End If
Label2.Caption = Label2.Caption + " " + Str(Round(s / k, 3))
End Sub

Private Sub CommandButton2_Click()
    TextBox1.Text = ""
    Label2.Caption = ""
    TextBox1.SetFocus
End Sub

Private Sub CommandButton3_Click()
    UserForm1.Hide
End Sub

```

Перехват и обработка ошибок

Каждая ошибка имеет свой код. Коды наиболее часто встречающихся ошибок приведены в таблице 6.1.

Таблица 6.1. Коды наиболее часто встречающихся ошибок

Код	Сообщение
5	Приложение не запущено
6	Переполнение
7	Не хватает памяти
9	Индекс выходит за пределы допустимого диапазона
11	Деление на нуль

13	Несоответствие типа
18	Произошло прерывание, вызванное пользователем
52	Неправильное имя файла или идентификатор
53	Файл не найден
54	Неверный режим работы с файлом
55	Файл уже открыт
56	Ошибка ввода-вывода
61	Переполнение диска
76	Путь не найден

В приложении следует создать средства перехвата любой возможной ошибки, обработать ее, выдать сообщение пользователю и обеспечить безаварийное продолжение приложения. Обычно конструкция перехвата ошибок имеет следующую структуру.

Инструкция On Error. Производит перехват ошибки, устанавливает, что программа должна делать в случае появления ошибки. Допустимы следующие синтаксисы.

Синтаксис 1: On Error Goto строка

Активизирует обработчик ошибок, начало которого определяется обязательным аргументом строки, значением которого может быть либо метка строки, либо номер строки. Обработчик ошибок определяет тип возникшей ошибки и устанавливает, что программа должна делать в зависимости от типа ошибки

Синтаксис 2: On Error Resume Next

Указывает, что при возникновении ошибки происходит передача управления на инструкцию, непосредственно следующую за инструкцией, где возникла ошибка.

Синтаксис 3: On Error Goto 0

Отключает любой активизированный обработчик ошибок в текущей процедуре.

Инструкция Resume. Обеспечивает процедуре возможность продолжить работу после обработки ошибки. Допустимы следующие синтаксисы.

Синтаксис 1: Resume

После обработки ошибки управление передается той инструкции, в которой произошла ошибка.

Синтаксис 2: Resume строка

После обработки ошибки управление передается инструкции, определенной аргументом строки. Значением этого аргумента может быть любая метка строки или номер строки.

Синтаксис 3: Resume Next

После обработки ошибки управление передается инструкции, следующей за инструкцией, в которой произошла ошибка.

Инструкция Exit. Останавливает выполнение процедуры. Допустимы синтаксисы:

- Exit Sub
- Exit Function
- Exit Property

Обработчик ошибок обычно включает объект Err, который содержит информацию об ошибках выполнения.

Свойства объекта Err:

Number	Возвращает код ошибки
Source	Имя текущего проекта VBA
Description	Возвращает строковое выражение, содержащее текст сообщения об ошибке
HelpFile	Полное имя файла справки VBA
HelpContext	Контекстный идентификатор файла справки VBA, соответствующий ошибке с кодом, указанным в свойстве Number

Основной метод объекта Err – Clear. Очищает все значения свойств объекта Err. Метод Clear используется для явной очистки значений свойств объекта Err после завершения обработки ошибки. Это необходимо, например, при отложенной обработке ошибки, которая задается инструкцией On Error Resume Next.

Рассмотрим на конкретном примере, как применяются инструкция On Error и объект Err при создании обработчика ошибок.

Пример. Создать программу, которая показывает фотографии однокурсников. Фамилии студентов выбираются из раскрывающегося списка ComboBox и при нажатии кнопки «Показать фотографию» фотография выбранного студента отображается в области рисунка. Для Image1 в свойство PictureSizeMode установить значение fmPictureSizeModeZoom. Фамилии студентов записаны на рабочем листе в столбце A, откуда они заносятся в раскрывающийся список

ComboBox. Полные имена файлов, содержащих фотографии записаны в столбце В, откуда они заносятся в список ListBox. Список ListBox сделать невидимым. При запуске программы в раскрывающемся списке должна отображаться фамилия первого в списке студента, а в области рисунка - его фотография. Предусмотреть обработку ошибочных ситуаций. Если имя файла или путь к файлу заданы неверно, вывести об этом сообщение (рис. 3.4).

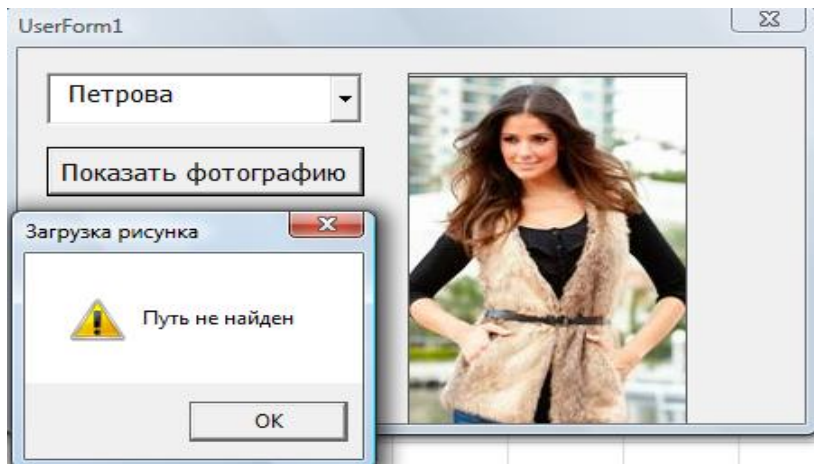


Рис. 3.4.

Код модуля формы

```
' Процедура обработки ошибочной ситуации
Sub Обработка_ошибки()
Select Case Err.Number
    Case 53
        MsgBox "Файл не найден", 48, "Загрузка рисунка"
    Case 76
        MsgBox "Путь не найден", 48, "Загрузка рисунка"
    Case Else
        MsgBox "Произошла ошибка: " & Err.Description
End Select
End Sub

' Обработка события Initialize объекта Userform
Private Sub userform_initialize()
' передача управления на обработчик ошибок,
```

```

' помеченный меткой строка1
On Error GoTo строка1
' Список ListBox делаем невидимым
ListBox1.Visible = False
With ListBox1
    .ColumnCount = 1
    .RowSource = ("B1:B20")
End With
With ComboBox1
    .ColumnCount = 1
    .RowSource = ("A1:A20")
    ' Отображение фамилии первого в списке студента
    .ListIndex = 0
End With
' Отображение фотографии первого в списке студента
Image1.Picture = LoadPicture(ListBox1.List(0, 0))
Exit Sub
' Обработчик ошибок
строка1:
' вызов процедуры обработки ошибок
Call Обработка_ошибки
End Sub

Private Sub CommandButton1_Click()
Dim i As Integer
' передача управления на обработчик ошибок,
' помеченный меткой Строка2
On Error GoTo Строка2
i = ComboBox1.ListIndex ' индекс выбранного элемента
If ListBox1.List(i, 0) = "" Then
    MsgBox "Не задано имя файла", 48, "Загрузка рисунка"
End If
Image1.Picture = LoadPicture(ListBox1.List(i, 0))
'Выход из процедуры при успешном выполнении загрузки рисунка
Exit Sub
' Обработчик ошибок
Строка2:
' вызов процедуры обработки ошибок

```

Call Обработка_ошибки
End Sub

Порядок выполнения программы.

Модуль формы состоит из трех процедур:

- процедура обработки ошибочной ситуации;
- процедура обработки события Initialize объекта Userform;
- процедура обработки события CommandButton1_Click.

Процедура Обработка_ошибки с помощью оператора выбора анализирует код ошибки. Если имя файла задано неверно, при загрузке рисунка возникает ошибка с кодом 53, если путь к файлу задан неверно - ошибка с кодом 76. Обработчик ошибок выводит соответствующее сообщение. При появлении других ошибок, выполнение программы прерывается, пользователь информируется об ошибке. Осуществляется выход из процедуры.

Процедура обработки события Initialize объекта Userform заполняет список ComboBox из диапазона A1:A20, в который предварительно введены фамилии студентов, список ListBox - из диапазона B1:B20 полными именами файлов. В раскрывающемся списке отображается фамилия первого в списке студента, а в области рисунка - его фотография. При возникновении ошибки при загрузке рисунка вызывается процедура Обработка_ошибки.

Процедура обработки события CommandButton1_Click фотографию выбранного из раскрывающегося списка студента отображает в области рисунка. Если при загрузке рисунка возникает ошибка, вызывается процедура Обработка_ошибки.

Задания для самостоятельного выполнения

1. В четырехугольнике ABCD $AB = x$, $BC = y$, $CD = z$, $DA = t$, $\angle ABC = \alpha$. Найти площадь четырехугольника, воспользовавшись функцией вычисления диагонали четырехугольника по формуле косинусов и процедурой вычисления площади треугольника по формуле Герона. Для ввода исходных данных и вывода результатов создать диалоговое окно с помощью формы.
2. V_1 , V_2 , V_3 объемы шаров, радиусы которых равны r_1 , r_2 , r_3 . Вычислить среднее арифметическое объемов шаров. Для вычисления объема шара создать пользовательскую функцию, передаваемый параметр – значение радиуса, возвращаемое

значение – объем шара. Объем шара вычисляется по формуле:

$V = \frac{4}{3} \pi R^3$. Для ввода радиусов на рабочем листе разместить три поля ввода TextVox, результаты записать в ячейки рабочего листа.

3. Используя процедуру определения наибольшего общего делителя двух чисел, сократить дроби $\frac{M}{N}$ и $\frac{P}{Q}$. Для ввода исходных данных и вывода результатов создать диалоговое окно с помощью формы.

4. Вычислить значение функции $z = \frac{s_1 + s_2}{k_1 * k_2}$. Здесь s_1, k_1 - сумма и число положительных элементов массива $x(n)$, s_2, k_2 - сумма и число положительных элементов массива $y(m)$. Воспользоваться процедурой, вычисляющей сумму и число положительных элементов массива. Для ввода элементов массивов на рабочем листе разместить два поля ввода TextVox, результат записать в ячейку рабочего листа.

5. Вычислить среднее геометрическое положительных элементов массивов $a(n)$, $b(m)$, $c(k)$. Использовать функцию, вычисляющую среднее геометрическое положительных элементов массива. Для ввода исходных данных и вывода результатов создать диалоговое окно с помощью формы.

6. Удельная теплоемкость газа при определенной температуре вычисляется стандартным выражением. Одна из принятых форм этого выражения представляет собой полином третьего порядка по температуре T : $C_p = a + bT + cT^2 + dT^3$. Если коэффициенты a, b, c, d для данного газа известны, можно вычислить его теплоемкость при любой температуре T . Создать пользовательскую функцию (параметры – значения температуры и четырех коэффициентов), которая возвращает вычисленную теплоемкость при данной температуре. Найти с помощью этой функции теплоемкость водяного пара при температуре 300°C, 500°C, 750°C. Для водяного пара значения коэффициентов:

$$a = 33,46 \times 10^{-3}, \quad b = 0,6880 \times 10^{-5}, \quad c = 0,7607 \times 10^{-8}, \quad d = -3,593 \times 10^{-12}$$

7. Основные объекты VBA в Excel

7.1. 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. Кроме того, объект Application позволяет вызывать более 400 встроенных функций рабочего листа при помощи конструкции вида:

Application.ФункцияРабочегоЛиста(Аргументы)

Например:

Application.Pi()	Вычисление числа π
Application.Pmt(Аргументы)	Определение постоянных периодических платежей при постоянной процентной ставке с помощью функции ППЛАТ (PMT)
Application.Sum(Аргументы)	Нахождение суммы значений из диапазона

Свойства объекта Application

ActiveWorkbook ActiveSheet ActiveCell ActiveChart ActiveDialog	Возвращает активный объект: рабочую книгу, лист, ячейку, диаграмму, диалоговое окно. В следующем примере в активной ячейке устанавливается полужирный шрифт и в нее выводится строка текста «Отчет за май»: With ActiveCell .Font.Bold = true .Value = "Отчет за май" End With
--	--

Методы объекта Application

Run	Запускает на выполнение подпрограмму или макрос. Синтаксис: Run (Macro, Arg1, Arg2, ...) <ul style="list-style-type: none">Macro – строка с именем макросаArg1, Arg2, ... – аргументы передаваемые макросу Например: Application.Run.Macro:= "Расчет"
-----	--

	Запускает макрос Расчет
Wait	<p>Временно приостанавливает работу приложения без остановки работы других программ. Синтаксис: Wait(Time)</p> <p>В следующем примере показывается, как установить время, чтобы возобновление работы приложения началось в 17 часов: Application. Wait "17:00:00"</p>
OnKey	<p>Устанавливает выполнение процедуры при нажатии комбинации клавиш. Синтаксис: OnKey (Key, Procedure)</p> <ul style="list-style-type: none"> ▪ Procedure - имя выполняемой подпрограммы при нажатии клавиш ▪ Key - строка, определяющая комбинацию клавиш, которая должна быть нажата. <p>Допустимо использование сочетания одновременно нажатых клавиш. С этой целью для перечисленных трех клавиш установлены следующие коды:</p> <ul style="list-style-type: none"> ▪ <Shift> - + ▪ <Ctrl> - ^ ▪ <Alt> - % <p>В следующем примере процедуре «Амортизация» назначена комбинация клавиш <Ctrl>+<+>, а процедуре «Процентная ставка» - <Shift>+<Ctrl>+<→>: Application. OnKey "^{+}", ""Амортизация" Application. OnKey "+^{RIGHT}", "Процентная ставка"</p>
Quit	Закрывает приложение.

7.2. Объект Workbook и семейство Workbooks

В иерархии Excel Объект Workbook (рабочая книга) идет сразу после объекта Application и представляет файл рабочей книги. Рабочая книга хранится либо в файлах формата XLSX (стандартная рабочая книга), XLSM (рабочая книга с поддержкой макросов), XLS (книга Excel97-2003) или XLA (полностью откомпилированное приложение).

Свойства объекта Workbook и семейства Workbooks

ActiveSheet	<p>Возвращает активный лист книги. Например: MsgBox "Имя активного листа" & ActiveSheet.Name</p> <p>- выводит в диалоговом окне имя активного рабочего</p>
-------------	--

	листа
ActiveChart	Возвращает активную диаграмму
Sheets	Возвращает семейство всех листов книги
Worksheets	Возвращает семейство всех рабочих листов книги
Count	Возвращает число объектов семейства Workbooks

Методы объекта Workbook и семейства Workbooks

Activate	Активизирует рабочую книгу так, что ее первый рабочий лист становится активным.
Add	Создает новую рабочую книгу.
Close	Закрывает рабочую книгу.
Open	Открывает существующую рабочую книгу.
Save	Сохраняет рабочую книгу.
SaveAs	Сохраняет рабочую книгу в другом файле. Синтаксис: SaveAs (Filename) <ul style="list-style-type: none"> ▪ Filename - строка, указывающая имя файла, под которым будет сохранена рабочая книга

7.3. Объект Worksheet и семейство Worksheets

В иерархии Excel объект Worksheet идет разу после объекта Workbook и представляет рабочий лист.

Свойства объекта Worksheet и семейства Worksheets

Name	Возвращает имя рабочего листа
UsedRange	Возвращает диапазон, т.е. объект Range, который содержит данные. В следующем примере очищается диапазон первого рабочего листа с данными: Worksheets(1). UsedRange.Clear
ActiveCell	Возвращает активную ячейку активного рабочего листа
Intersect	Возвращает диапазон, являющийся пересечением нескольких диапазонов. Синтаксис: Intersect (range1, range2, ...) В следующем примере выбирается пересечение диапазонов A1:D3 и C3:D4, т.е. диапазон C3:D3 Intersect (Range("A1:D3"), Range ("C3:D4")).Select
Union	Возвращает диапазон, являющийся объединением

	<p>нескольких диапазонов. Синтаксис: Union (range1, range2, ...) В следующем примере выбирается объединение двух диапазонов A1:B2 и C3:D4 Union (Range("A1:B2"), Range ("C3:D4")).Select</p>
--	---

Методы объекта **Worksheet** и семейства **Worksheets**

Activate	Активизирует указанный рабочий лист
Add	<p>Создает новый рабочий лист. Синтаксис: Add (Before, After, Count, Type)</p> <ul style="list-style-type: none"> ▪ Before - указывает лист, перед которым будет размещен новый рабочий лист ▪ After - указывает лист, после которого будет размещен новый рабочий лист. Если аргументы Before и After опущены, то новый лист размещается перед активным листом ▪ Count - число добавляемых листов, по умолчанию 1 ▪ Type - указывает тип добавляемого листа. <p>Допустимые значения: x1Worksheet (по умолчанию), x1ExcelMacroSheet, x1ExcelIntMacroSheet</p> <p>Например: ActiveWorkbook. Worksheets.Add - вставляет новый лист перед активным листом активной рабочей книги</p>
Delete	<p>Удаляет рабочий лист. Например: Worksheets(1). Delete - удаляет первый рабочий лист из активной рабочей книги</p>
Copy	<p>Копирует рабочий лист в другое место рабочей книги. Синтаксис: Copy (Before, After)</p> <ul style="list-style-type: none"> ▪ Before - рабочий лист перед которым вставляется данный ▪ After - рабочий лист после которого вставляется данный <p>Одновременно допустимо использование только одного из аргументов. В следующем примере «Лист1» активной</p>

рабочей книги копируется после «Лист3» той же рабочей книги: Worksheets("Лист1"). Copy After:= Worksheets("Лист3")

7.4. Объекты Range и Selection

В иерархии Excel объект Range 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. Интересной особенностью объектов *Range* и *Selection* является то, что они не являются элементами никакого семейства объектов.

Адресация ячеек

При работе с объектом *Range* необходимо помнить, как в Excel ссылаются на ячейку рабочего листа. Имеются два способа ссылки на ячейки рабочего листа: относительная адресация и абсолютная адресация. Относительная адресация:

Формат A1	Имя ячейки состоит из имени столбца (их 256 – A, B, . . . Z, AA, AB, . . . IV) и номера строки (1, . . . 16384)
Формат R1C1	Адресация задается индексом строки и индексом столбца. Например, R2C3, R4C5

Абсолютная адресация:

Формат A1	Признаком абсолютной адресации является знак доллара \$, который записывается перед именем строки или столбца. Например, \$A10, A\$10, \$A\$10
Формат R1C1	Указывается смещение по отношению к активной ячейке. Смещение записывается в квадратных скобках, знак указывает на направление смещения. Например, если R2C3 – активная ячейка, то R[1]C[-1] дает ссылку на ячейку R3C2.

Адресация ячейки рабочего листа является лишь частью полного адреса ячейки, который в общем случае включает имя рабочего листа и адрес книги. При задании полного адреса за именем листа пишется восклицательный знак !, а адрес книги заключается в скобки. Например,

A1
Лист2!A1

Задание групп строк и столбцов

Если в диапазоне указывается 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, B, C, а Range("2:2") – из второй строки. Другим способом работы со строками и столбцами являются методы Rows (строки) и Columns (столбцы). Например столбцом A является Columns(1), а второй строкой Rows(2).

Связь объекта Range и свойства Cells

Ячейка является частным случаем диапазона, состоящий только из единственной ячейки, объект Range позволяет работать с ячейкой. Объект Cells (ячейки) – это способ работы с ячейкой. Например, ячейка A2 как объект описывается Range("A2") или Cells(1,2). В свою очередь, Cells вкладываясь в Range, также позволяет записывать диапазон. Range("A2:C3") и Range(Cells(1,2), Cells(3,3)) определяют один и тот же диапазон.

Основные свойства объекта Range

Value	Возвращает значение ячейки или в ячейку диапазона. В данном примере переменной x присваивается значение из ячейки C1: x = Range("C1").Value
Name	Возвращает имя диапазона. В данном примере диапазону A1:B2 присваивается имя Итоги: Range("A1:B2"). Name="Итоги"
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
Formula	Возвращает формулу в формате A1. Например, следующая инструкция вводит в ячейку C2 формулу =\$A\$4+\$A\$10: Range("C2"). Formula = "=\$A\$4+\$A\$10"
FormulaLocal	Возвращает неанглоязычные (местные) формулы в формате A1. Например, следующая инструкция

	вводит в ячейку B2 формулу =СУММ(C1:C4) : Range("B2"). FormulaLocal = "=СУММ(C1:C4)"
FormulaR1C1	Возвращает формулу в формате R1C1. Например, Range("B2"). Formula R1C1 = "=SQR(R3C2) "
FormulaR1C1Local	Возвращает неанглоязычные (местные) формулы в формате R1C1.
Text	Возвращает содержание диапазона в текстовом формате.

Основные методы объекта Range

Adress	<p>Возвращает адрес ячейки. Синтаксис: Adress(rowAbsolute,colomnAbsolute,referencesStyle, external, relativeTo)</p> <p>Аргументы:</p> <ul style="list-style-type: none"> ▪ rowAbsolute – допустимы два значения True и False. Если True или аргумент опущен, то возвращается абсолютная ссылка на строку; ▪ colomnAbsolute – допустимы два значения True и False. Если True или аргумент опущен, то возвращается абсолютная ссылка на столбец; ▪ referencesStyle - допустимы два значения x1A1 и x1R1C1, если используется значение x1A1 или аргумент опущен, то возвращается ссылка в виде формата A1; ▪ external - допустимы два значения True и False. Если False или аргумент опущен, то возвращается относительная ссылка; ▪ relativeTo – в случае, если rowAbsolute и colomnAbsolute равны False, а referencesStyle - x1R1C1, то данный аргумент определяет начальную ячейку диапазона, относительно которой производится адресация. <p>Следующий пример показывает различные результаты адресации.</p> <p>MsgBox Cells(1, 1). Adress() ' В диалоговом окне отображается адрес \$A\$1</p> <p>MsgBox Cells(1, 1). Adress(rowAbsolute:=False) ' В диалоговом окне отображается адрес \$A1</p> <p>MsgBox Cells(1, 1). Adress(referencesStyle:= x1R1C1)</p>
--------	--

	' В диалоговом окне отображается адрес R1C1
Clear	Очищает диапазон
Copy	Копирует диапазон в другой диапазон или в буфер обмена. Синтаксис: Copy (destination) В данном примере диапазон A1:D4 рабочего листа Лист1 копируется в диапазон E5:H8 листа Лист2: Worksheets("Лист1").Range("A1:D4").copy _ destination:= Worksheets("Лист2").Range("E5")
Columns Rows	Возвращает количество столбцов (строк), из которых состоит диапазон I=Selection. Columns.Count J= Selection. Rows.Count
Delete	Удаляет диапазон Rows(3).Delete
Insert	Вставка ячейки или диапазона ячеек Worksheets("Лист1").Rows(4).Insert Вставляется новая строка перед четвертой строкой рабочего листа Лист1
Offset	Возвращает диапазон, смещенный относительно данного диапазона на величины указанные в аргументах. Синтаксис: Offset(rowOffset, columnOffset) Аргументы: rowOffset – целое число, указывающее сдвиг по строкам; columnOffset – целое число, указывающее сдвиг по столбцам. Например, в следующем примере активизируется ячейка, расположенная на три строки ниже и на два столбца левее относительно предыдущей активной ячейки: ActiveCell.Offset(3,-2). Activate
Select	Выделение диапазона

7.5. Объект ActiveCell

Изменение значения активной ячейки

Объект ActiveCell указывает на месторасположение активной ячейки, а также предоставляет доступ к ее содержимому и к ее свойствам. Чтобы получить доступ к содержимому активной ячейки

или изменить ее, используется свойство `ActiveCell.Value`. Например, если значение активной ячейки больше 5, поместим в нее значение 100, иначе в ячейку поместим строку «меньше пяти».

```
If ActiveCell.Value >= 5 Then
    ActiveCell.Value = 100
Else
    ActiveCell.Value = "меньше пяти"
EndIf
```

Изменение свойств активной ячейки

Для изменения шрифта активной ячейки используется свойство – `Font`.

`ActiveCell.Font.Color` или `.ColorIndex` – изменяет цвет шрифта; `Color=RGB(red, green, blue)`, где `red`, `green`, `blue` значения интенсивностей красного, зеленого и синего цвета, изменяющиеся в пределах от 0 до 255. `ColorIndex` – целое число в интервале от 0 до 255. Например, `ActiveCell.Font.Color = RGB(0,255,0)` устанавливает зеленый цвет.

`ActiveCell.Font.Name` – имя шрифта, заключенное в кавычки, например, “Courier”.

`ActiveCell.Font.Size` – размер шрифта, например,
`ActiveCell.Font.Size = 24`.

Изменение положения активной ячейки

Свойство `Offset` объекта `ActiveCell` позволяет выбирать ячейку, которая расположена со сдвигом на указанное количество строк или столбцов относительно активной ячейки. Метод `Activate` делает выбранную ячейку активной.

`Offset(rowOffset, columnOffset)` – здесь:

`rowOffset` – целое число, указывающее сдвиг по строкам;

`columnOffset` – целое число, указывающее сдвиг по столбцам.

Использование значений близлежащих ячеек для вычисления значения активной ячейки

Свойство `Offset` также используется для захвата значений из близлежащих ячеек и для выполнения вычислений.

Пример 1. Вычислить значения функции $y=\sin x$, при x изменяющемся от 0 до 180° с шагом 30° , результаты вычислений записать на рабочий лист (рис. 7.3).

Код модуля рабочего листа Лист1

```
Private Sub CommandButton1_Click()
Const pi = 3.14
Dim x, dx As Integer
Dim y As Double
' в dx записываем значение фиксированной ячейки C1 –
' шаг изменения аргумента x
x = 0: dx = Cells(1, 3).Value
' ячейку B2 делаем активной
Cells(2, 2).Activate
For x = 0 To 180 Step dx
' в ячейку расположенной слева от активной записываем
значение x
ActiveCell.Offset(0, -1).Value = x
y = Sin(x * pi / 180)
' в активную ячейку записываем значение y
ActiveCell.Value = y
' смещаем активную ячейку на одну строку вниз
ActiveCell.Offset(1, 0).Activate
Next
End Sub
```

	A	B	C	D
1	x	y=sinx	30	
2	0	0,0000		
3	30	0,4998		
4	60	0,8658		
5	90	1,0000		
6	120	0,8666		
7	150	0,5011		
8	180	0,0016		
9				
10				

Вычислить

Рис. 7.3.

Пример 2. На листе Excel записаны фамилии студентов и экзаменационные оценки за первый семестр (рис. 7.4). Подсчитать среднюю оценку, и студентам, имеющим средний балл не ниже четырех начислить стипендию. Размер стипендии записан в ячейке I2 рабочего листа.

	A	B	C	D	E	F	G	H	I	
1	Назначение стипендии									
2							Размер стипендии		1100	
3	Фамилия	Информ.	Матем.	Англ.яз.	Средняя оценка	Стипендия				
4										
5	Алексеева	3	4	5	4,00	1100				
6	Гарипова	4	3	4	3,67					
7	Гунина	4	4	5	4,33	1100				
8	Крылова	3	3	4	3,33					
9	Ткаченко	4	4	4	4,00	1100				
10	Шубина	5	4	5	4,67	1100				
11										

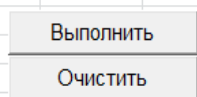


Рис. 7.4.

Код модуля рабочего листа Лист1

```
Dim n, i, j As Integer, st As Single
Private Sub CommandButton1_Click()
st = Range("I2").Value
' Определяем число студентов в списке
n = Application.CountA(ActiveSheet.Columns(1)) - 2
' ячейку E5 делаем активной
Cells(5, 5).Activate
For i = 0 To n - 1
' В активную ячейку записываем формулу
' для вычисления средней оценки
ActiveCell.FormulaR1C1 = "=Round(SUM(RC[-3]:RC[-1])/3,2)"
' если средняя оценка не меньше четырех, студенту начисляем
стипендию
If ActiveCell.Value >= 4 Then
ActiveCell.Offset(0, 1).Value = st
Else
ActiveCell.Offset(0, 1).Value = " "
End If
' активную ячейку смещаем на одну строку вниз
ActiveCell.Offset(1, 0).Activate
```



```

Next
End Sub

Private Sub CommandButton2_Click()
m = n + 5
Range(Cells(5, 5), Cells(m, 6)).Clear
End Sub

```

Диапазон и массив

В VBA имеется тесная связь между диапазонами и массивами. Допустимо как заполнение массива одним оператором присваивания значениями из ячеек диапазона, так и наоборот, заполнение диапазона ячеек одним оператором присваивания элементами массива. При этом массив должен быть объявлен как переменная типа Variant.

Пример инициализации массива из диапазона одним оператором присваивания:

```

Dim r As Range
Set r = Range("A1:D3")
Dim m As Variant
Dim i, j As Integer
m = r.Value

```

Пример заполнения диапазона из массива посредством одного оператора присваивания:

```

Dim r As Range
Dim m(1 To 9, 1 To 9)
Dim i, j As Integer
For i = 1 To 9
  For j = 1 To 9
    m(i, j) = i * j
  Next
Next
Set r = Range(Cells(1, 1), Cells(9, 9))
r.Value = m

```

Пример 1. На рабочем листе Excel дан числовой массив $a(n, n)$. Найти сумму элементов расположенных ниже главной диагонали. Сумму вывести на надпись (рис.7.6).

	A	B	C	D	E	F	G	H	I
1	12	23	18	24	61	45		s= 871	
2	33	56	28	27	4	63			
3	81	35	101	56	23	66			
4	80	67	73	44	15	12		Вычислить	
5	45	34	56	43	82	31			
6	92	27	22	78	105	67		Очистить	

Рис. 7.6.

Код модуля рабочего листа Лист1

```

Private Sub CommandButton1_Click()
Dim r As Range
Dim a As Variant
Dim i, j, n As Integer
Dim s As Single
' r присваиваем диапазон, содержащий данные
Set r = Worksheets(1).UsedRange
' Определяем число строк диапазона
n = r.Rows.Count
a = r.Value
' Вычисление суммы элементов расположенных ниже главной
диагонали
s = 0
For i = 1 To n
    For j = 1 To n
        If j < i Then s = s + a(i, j)
    Next j
Next i
Label2.Caption = "s= " + Str(s)
End Sub

Private Sub CommandButton2_Click()
Label2.Caption = ""
Worksheets(1).UsedRange.Clear
End Sub

```

Пример 2. На рабочем листе Excel дан числовой массив $a(n, m)$. Элементы, кратные 7, увеличить на значение их индекса. Измененный

массив вывести на рабочий лист на две строки ниже относительно исходного массива (рис. 7.7).

	A	B	C	D	E	F	G
1		Исходный массив					
2	-7	63	45	66		Выполнить	
3	23	-28	72	91			
4	-5	77	-33	12		Очистить	
5	55	-77	35	87			
6	-32	28	63	22			
7							
8		Измененный массив					
9	-5	66	45	66			
10	23	-24	72	97			
11	-5	82	-33	12			
12	55	-71	42	87			
13	-32	35	71	22			

Рис. 7.7.

Код модуля рабочего листа Лист1

```

Private Sub CommandButton1_Click()
Dim r As Range
Dim a As Variant
Dim i, j, n, m As Integer
Set r = ActiveSheet.UsedRange
' Определяем число строк и столбцов диапазона
n = r.Rows.Count
m = r.Columns.Count
a = r.Value
For i = 1 To n
    For j = 1 To m
        If a(i, j) Mod 7 = 0 Then a(i, j) = a(i, j) + i + j
    Next j
Next i
Set r = Range(Cells(n + 4, 1), Cells(2 * n + 3, 1 + m - 1))
r.Value = a
' На рабочий лист выводим названия массивов
Range("B1").Value = "Исходный массив"
Range("B" + Trim(Str(n + 3))).Value = "Измененный массив"
End Sub

```

```
Private Sub CommandButton2_Click()  
ActiveSheet.UsedRange.Clear  
End Sub
```

Задания для самостоятельного выполнения

1. На рабочем листе Excel дан числовой массив a (n,m). Подсчитать сумму и количество всех четных элементов массива. Ответ вывести на надпись.
2. На рабочем листе Excel дан числовой массив a (n,m). Все положительные элементы массива увеличить на 20, а отрицательные на 5. Измененный массив вывести на рабочий лист на три строки ниже относительно исходного массива.
3. На рабочем листе Excel дан числовой массив a (n,m). Все элементы, расположенные в четных столбцах, заменить полусуммой индексов этих элементов. Измененный массив вывести на рабочий лист.
4. На листе Excel записаны фамилии спортсменов-многоборцев и количество очков, набранных ими по 5 видам спорта. Подсчитать общее количество очков для каждого спортсмена. Если спортсмен в сумме набрал не менее k очков, то ему присваивается звание мастера. Общее количество очков и отметку о присваивании спортсмену звания мастера спорта записать в последующие столбцы рабочего листа.
5. На листе Excel записаны фамилии жильцов и продолжительность разговора по телефону за текущий месяц в минутах. Подсчитать плату услуг телефонной связи для каждого клиента, если телефонная компания взимает плату по следующему тарифу: 370 мин в месяц оплачиваются как абонентская плата, которая составляет 280 рублей в месяц. За каждую минуту сверх нормы необходимо платить по 0,25 рублей. Подсчитать сумму оплаты и записать в следующий столбец рабочего листа.

8. Методы объекта Range, использующие команды Excel

8.1. Метод GoalSeek

При решении задач полезно бывает анализировать, как меняется значение формулы при изменении значения одной из переменных, и при этом чаще всего требуется отыскать наилучшее, оптимальное значение данного параметра. Например, при каком объеме тиража

можно будет покрыть все расходы на издательство книги и начинать получать прибыль. Метод GoalSeek (команда «Подбор параметра») позволяет отыскать приводящее к нужному результату значение переменной. Этот метод также можно использовать для анализа функций, решения уравнений. Говоря простым математическим языком, *подбор параметра* – это средство отыскания решения уравнения с одним неизвестным.

Метод GoalSeek 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.

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

Формула, которая использует в качестве аргумента данные, зависящие от ее собственного значения, называется формулой с *циклическими ссылками*. Иногда такая ситуация возникает в результате какой-нибудь ошибки, однако некоторые задачи требуют для своего решения использования именно циклических ссылок. Для решения подобных задач Excel использует итерации, т.е. повторяющиеся циклические вычисления, выполняемые до тех пор, пока не будет выполнено требуемое условие.

Для поиска решения метод GoalSeek использует *метод итераций*. Это означает следующее: прежде всего проверяется начальное значение в изменяемой ячейке (ChangingCell), содержащей параметр. Если начальное значение не дает требуемое значение (Goal) целевой функции, производится изменение значения параметра и новое

вычисление результата, и так до тех пор, пока не будет отыскано нужное значение параметра (если, конечно, итерации сходятся).

Точность, с которой находится корень и предельно допустимое число итераций, используемых для нахождения корня, устанавливаются 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
```

Пример 1. Создать программу приближенного решения уравнения $(2x^2+3)(1-\sin x)-\ln x=2$. Результаты вычислений вывести на рабочий лист Excel.

Для решения уравнения используем метод GoalSeek, который вычисляет корень, используя метод последовательных приближений, результат выполнения которого зависит от начального приближения.

Поместим в ячейку B5 начальное значение x , например, единицу. В ячейку B3 поместим значение правой части уравнения (в нашем примере это два). (Рис. 8.1).

	A	B	C	D
1	Решение алгебраического уравнения $(2x^2+3)(1-\sin x)-\ln x=2$			
2				
3	Значение правой части	2		
4	Значение левой части			
5	Значение x	1		
6				
7				
8				

Рис. 8.1.

Напишем процедуру обработки события Click для CommandButton1 (кнопка «Выполнить»). В ячейку B4 вставим формулу вычисления левой части уравнения. Применим метод GoalSeek: в ячейке B4 установить значение 2, изменяя значение ячейки B5, т.е. напишем инструкцию:

```
Range("B4").GoalSeek goal:=2, CchangingCell:=Range("B5")
```

После нажатия на кнопку «Выполнить» результат расчета будет помещен в ячейки B5 (значение корня в данном случае 0,635796191) и B4 (значение левой части уравнения при найденном значении корня) в

данном случае оно равно 1,999809434) (рис. 8.2). В силу того решение находится приближенно с указанной точностью в ячейке B4 получилось 1,999809434, а не ровно 2. Увеличивая точность можно найти лучшее приближение к корню.

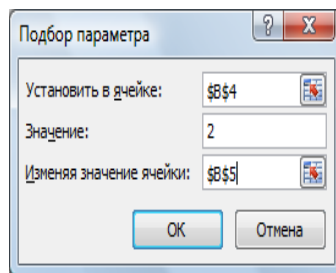
Код модуля рабочего листа Лист1

```
Private Sub CommandButton1_Click()
Range("B4").FormulaLocal = "=(2*B5^2+3)*(1-sin(B5))-ln(B5)"
Range("B4").GoalSeek goal:=2, changingCell:=Range("B5")
End Sub
```

	A	B	C	D
1	Решение алгебраического уравнения $(2x^2+3)(1-\sin x)-\ln x=2$			
2				
3	Значение правой части	2		
4	Значение левой части	1,999809434		
5	Значение x	0,635796191		
6				
7				
8				

Рис. 8.2.

В MS Excel 2010 данная задача решается следующим образом. После ввода в ячейки B3, B5 исходных данных и соответствующей формулы в ячейку B4, перейти на вкладку **Данные**, в группе **Работа с данными** из списка **Анализ "что-если"** выбрать команду **Подбор параметра**.



В окне **Подбор параметра** установить соответствующие значения и нажать на **ОК**.

Пример 2. Клиент хочет открыть счет в банке под 5% годовых и за пять лет накопить \$150 тыс. Он собирается производить ежегодные отчисления на этот счет и хочет узнать необходимые размеры ежегодного отчисления.

Для решения этой задачи можно использовать финансовую функцию **БС** и метод **GoalSeek**. Функция **БС** возвращает будущее значение вклада на основе периодических постоянных платежей и постоянной процентной ставки. Синтаксис этой функции:

БС(ставка; кпер; плт; пс; тип)

Эта функция имеет три обязательных параметра – процентную ставку, общее число периодов выплат и размер одиночной выплаты. Необязательные параметры:

пс — это приведенная к текущему моменту стоимость или общая сумма, которая на текущий момент равноценна ряду будущих платежей. Если аргумент пс опущен, то он полагается равным 0. В этом случае должно быть указано значение аргумента плт.

Тип — число 0 или 1, обозначающее, когда должна производиться выплата. Если аргумент «тип» опущен, то он полагается равным 0.

Тип	Когда нужно платить
0	В конце периода
1	В начале периода

В ячейку В2 поместим процентную ставку, в ячейку В3 – количество периодов выплат (рис. 8.2). В ячейку В4 поместим формулу для вычисления размера одиночной выплаты: `Range("B5").FormulaLocal = "=БС(В2;В3;В4)"`

Применим метод GoalSeek: в ячейке В5 установить значение 150000, изменяя значение ячейки В4, т.е. напишем инструкцию:

`Range("B5").GoalSeek goal:=150000, changingCell:=Range("B4")`

После нажатия на кнопку «Вычислить» результат расчета будет помещен в ячейки В4 размер ежегодного отчисления - \$27146,22 и в ячейку В5 будущее значение вклада \$150000. (Рис. 8.3).

	А	В
1	Отыскание будущего значения вклада	
2	Процентная ставка	0,05
3	Количество периодов	5
4	Выплата	-\$27 146,22
5	Будущее значение вклада	\$150 000,00
6		
7	Вычислить	
8		

Рис. 8.3.

Код модуля рабочего листа Лист1

```
Private Sub CommandButton1_Click()  
Range("B5").FormulaLocal = "=БС(В2;В3;В4)"
```



```
Range("B5").GoalSeek goal:=150000, changingCell:=Range("B4")
End Sub
```

8.2. Метод AutoFill

Метод AutoFill (автозаполнение) автоматически заполняет ячейки диапазона элементами последовательности. При этом методе 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 протаскивании маркера заполнения вдоль диапазона, в котором будет располагаться создаваемая последовательность.

Синтаксис:

Объект.AutoFill(destination, type)

Аргументы:

Объект	Диапазон, с которого начинается заполнение
destination	Диапазон, который заполняется
type	Допустимые значения : x1FillDefault, x1FillSries, x1FillCopy, x1FillFormats, x1FillValues, x1FillDays, x1FillWeekDays, x1FillMonths, x1FillYears, x1LinearTrend, x1GrowthTrend. По умолчанию x1FillDefault

Пример 1. Создать программу заполнения таблицы значений функции $y = x^2 - 8$ при x изменяющемся от -3 до 3 с шагом 1.

	А	В
1	x	$y=x^2-8$
2	-3	1
3	-2	-4
4	-1	-7
5	0	-8
6	1	-7
7	2	-4
8	3	1

Рис. 8.4.

Сначала задачу решим на рабочем листе вручную. В ячейку A2 введем начальное значение x – минус три, в ячейку A3 следующее значение - минус два. Выделим диапазон A2:A3, содержащий два


первых значения x . Протащим маркер заполнения вниз по столбцу так, чтобы создать требуемую последовательность значений x , до ячейки A8. В ячейку B2 введем формулу вычисления значения y : $=A2^2-8$. Выделим данную ячейку и протащим маркер заполнения до ячейки B8 (рис. 8.4).

На VBA тот же результат получается следующим образом: для вычисления значения x аргументу `destination` присваивается значение `Range("A2:A8")`, аргументу `type` присваивается `x1FillDefault`, а метод применяется к диапазону `Range("A2:A3")`. Для вычисления значения y аргументу `destination` присваивается значение `Range("B2:B8")`, аргументу `type` присваивается `x1FillDefault`, а метод применяется к диапазону `Range("B2")`.

Таким образом, имеем:

```
Range("A2:A3").AutoFill Destination:=Range("A2:A8"), _
Type:=x1FillDefault
Range("B2").FormulaLocal = "=A2^2-8"
Range("B2").AutoFill Destination:=Range("B2:B8"), _
Type:=x1FillDefault
```

Пример 2. Создать программу, которая вычисляет n -ый член и сумму членов арифметической прогрессии, при n , изменяющемся от 1 до 10. Первый член арифметической прогрессии равен -2 , $d = 0,725$, n -ый член вычисляется по формуле: $a_n = a_1 + d(n-1)$, сумма : $s_n = (a_1 + a_n) * n / 2$.

На рабочем листе запишем исходные данные. (Рис. 8.5). В ячейку A5 запишем значение разности d . Ячейке A5 присвоим имя d , для этого выделим эту ячейку, перейдем на вкладку **Формулы** и в группе **Определение имени** нажмем кнопку **Присвоить имя** , откроется окно **Создание имени**, в поле **Имя** введем d . В столбец B начиная с пятой строки запишем номера первого и второго членов прогрессии, в ячейку C5 – первый член прогрессии, в ячейку D5 – S_1 . Применим метод `AutoFill` к диапазону `Range("B5:B6")`, аргументу `destination` присвоим значение `Range("B5:B14")`. В ячейку C6 запишем формулу вычисления второго члена прогрессии в формате R1C1. Применим метод `AutoFill` к ячейке `Range("C6")`, аргументу `destination` присвоим значение `Range("C6:C14")`. В ячейку D6 запишем формулу вычисления суммы двух первых членов прогрессии в

формате R1C1. Применим метод AutoFill к ячейке Range("D6"), аргументу destination присвоим значение Range("D6:D14").

	A	B	C	D	E	F
1	Вычисление n - го члена					
2	арифметической прогрессии					
3						
4		n	an	Sn		
5	0,725	1	-2	-2		
6		2	-1,275	-3,275		
7		3	-0,55	-3,825		
8		4	0,175	-3,65		
9		5	0,9	-2,75		
10		6	1,625	-1,125		
11		7	2,35	1,225		
12		8	3,075	4,3		
13		9	3,8	8,1		
14		10	4,525	12,625		

Рис.8.5.

Код модуля рабочего листа Лист1

```
Private Sub CommandButton1_Click()
Dim d As Single
d = Range("A5").Value
Range("B5:B6").AutoFill Destination:=Range("B5:B14")
Cells(6, 3).Activate
ActiveCell.FormulaR1C1 = "=R[-1]C+d"
Range("C6").AutoFill Destination:=Range("C6:C14")
Cells(6, 4).Activate
ActiveCell.FormulaR1C1 = "=(R5C3+RC[-1])*RC[-2]/2"
Range("D6").AutoFill Destination:=Range("D6:D14")
End Sub
```

8.3. Метод Sort

Сортировка позволяет выстраивать данные в алфавитном или цифровом порядке по возрастанию или по убыванию. Microsoft Excel может сортировать строки, а также столбцы списков рабочих листов. При сортировке текста в таблице можно сортировать как один столбец, так и таблицу целиком. Кроме того можно выполнить сортировку по нескольким словам или полям в одном столбце таблицы, а также - для выделенного диапазона списка. Если на

вкладке **Данные**, в группе **Сортировка и фильтр** выбрать кнопку **Сортировка**, то откроется окно, в котором можно задать ключи сортировки (столбцы или строки), порядок сортировки и некоторые другие параметры (рис. 8.6).

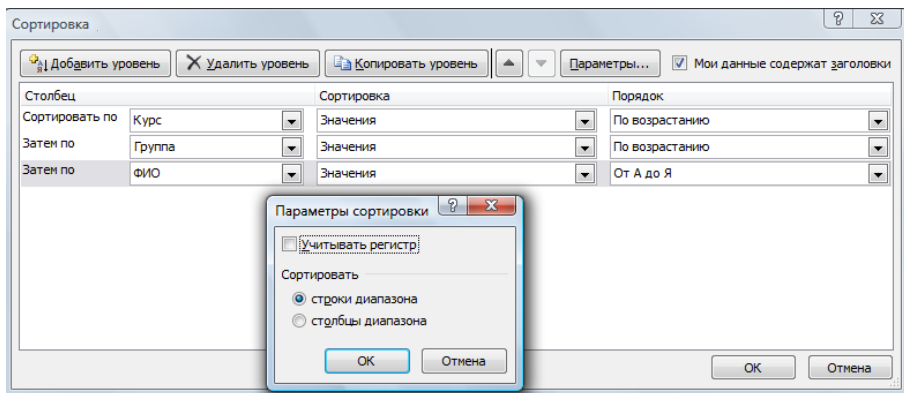


Рис. 8.6. Окно **Сортировка** с открытым окном **Параметры сортировки**

В VBA для осуществления сортировки данных с учетом до трех критериев, по которым производится сортировка, применяется метод `Sort`, который позволяет сортировать строки списков, сводных таблиц и баз данных, а также столбцы рабочих листов.

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: <ul style="list-style-type: none"> ▪ x1Ascending (возрастающий порядок) ▪ x1Descending (убывающий порядок)
key2	Ссылка на второе упорядочиваемое поле
order2	Задаёт порядок упорядочивания. Допустимые значения: <ul style="list-style-type: none"> ▪ x1Ascending (возрастающий порядок) ▪ x1Descending (убывающий порядок)

Key3	Ссылка на третье упорядочиваемое поле
Order3	Задаёт порядок упорядочивания. Допустимые значения: <ul style="list-style-type: none"> ▪ x1Ascending (возрастающий порядок) ▪ x1Descending (убывающий порядок)
header	Допустимые значения: <ul style="list-style-type: none"> ▪ x1Yes (первая строка диапазона содержит заголовок, который не сортируется) ▪ x1No (первая строка диапазона не содержит заголовок) ▪ x1Guess (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: <ul style="list-style-type: none"> ▪ x1TopToBottom (сортировка осуществляется сверху вниз, т.е. по строкам), ▪ x1LeftToRight (по столбцам)

8.4. Метод AutoFilter

Для поиска и фильтрации данных в MS Excel существуют автофильтр и расширенный фильтр. *Автофильтр* используется для простых условий отбора; а *расширенный фильтр* – для более сложных условий отбора.


В отфильтрованном списке отображаются только те строки, отвечающие условиям отбора, заданным для столбца. При фильтрации порядок записей в списке не изменяется, строки, которые не удовлетворяют критерию фильтрации, временно скрываются.

При использовании автофильтра для поиска данных в списке, справа от подписей столбцов в фильтруемом списке появляются стрелки автофильтра . У отфильтрованных данных номера строк окрашиваются в синий цвет, а стрелка автофильтра превращается в стрелку с воронкой (фильтром) , возле тех полей, по которым произведен отбор данных.

	A	B	C	D	E	F	G	H	I	J	K	
1	Социальная стипендия											
2											Прожиточный минимум	
3	№	Фамилия	Курс	Группа	Сред. З/п отца	Сред. З/п матер	Число членов семьи	Доход на 1 члена семьи	Социальная стипендия		Размер социальной стипендии	
4	11	Ахтареева	3	991	106							
5	12	Аюпова	3	991	68						Число заполненных строк	
6	20	Курбонов	3	991	151							
7	21	Митрофанова	3	991	74							
8	28	Мосолов	3	991	81							
9	13	Мухуддинова	3	991	156							
10	14	Рахметов	3	991	181							
11	10	Сапахиев	3	991	156							
12	24	Сафиуллин	3	991	152							
13	15	Шайхулislamов	3	991	151							
14	22	Шакирова	3	991	91							
15	23	Шарифзода	3	991	256							
16	16	Ахмадиев	5	971	131							
17	17	Гапеева	5	971	89							
18	24	Гафуров	5	971	151							
19	18	Давлетшина	5	971	91							
20	19	Ефименко	5	971	106							
21	20	Закирова	5	971	106							
22	21	Капимуллина	5	971	106							
23	27	Капитонова	5	971	101							
24	26	Миргазизова	5	971	91							
25	22	Мухаметзянова	5	971	156							
26	25	Низаев	5	971	141							
27	23	Хуснуллина	5	971	14500	6500	4	5250,00	0			

Рис. 8.7. Раскрывающийся список автофильтра

Для фильтрации данных с помощью автофильтра необходимо выполнить следующие действия.

1. Установить указатель ячейки в список данных.
2. Перейти на вкладку **Данные**, в группе **Сортировка и фильтр** использовать команду **Фильтр**. Возле каждого поля строки заголовка списка появятся стрелки автофильтра .
3. Перейти к необходимому полю.
4. Из раскрывающегося списка автофильтра выбрать требуемый критерий поиска (рис. 8.7).

Метод `AutoFilter` (автофильтр) объекта `Range` позволяет программным способом задать выполнение автофильтрации списка по критериям, указанным в параметрах. При применении метода `AutoFilter` допустимы два синтаксиса.

Синтаксис 1:

Объект. `AutoFilter`

В этом случае метод `AutoFilter` выбирает или отменяет команду **Данные, Фильтр** примененную к диапазону, заданному в аргументе Объект.

Синтаксис 2:

Объект. `AutoFilter(field,criteria1,operator, criteria2)`

В этом случае метод `AutoFilter` выполняет команду *Данные, Фильтр* по критериям, указанным в аргументах.

Аргументы:

Объект	Диапазон
<code>field</code>	Целое, указывающее поле, в котором производится фильтрация данных
<code>criteria1</code> и <code>criteria2</code>	Задают два возможных условия фильтрации поля. Допускается использование строковой постоянной, например 101, и знаковых отношений <code>></code> , <code><</code> , <code>>=</code> , <code><=</code> , <code>=</code> , <code><></code>
<code>operator</code>	Допустимые значения: <ul style="list-style-type: none"> ▪ <code>x1And</code> (логическое объединение первого и второго критериев) ▪ <code>x1Or</code> (логическое сложение первого и второго критериев) ▪ <code>x1Top10Items</code> (для показа первых десяти элементов поля)

При работе с фильтрами полезны метод `ShowAllData` и свойства `FilterMode` и `AutoFilterMode`.

Метод <code>ShowAllData</code>	Показывает все отфильтрованные и неотфильтрованные строки рабочего листа
Свойство <code>FilterMode</code>	Допустимые значения: <code>True</code> (если на рабочем листе имеются отфильтрованные данные со скрытыми строками), <code>False</code> (в противном случае)
Свойство <code>AutoFilterMode</code>	Допустимые значения: <code>True</code> (если на рабочем листе выведены раскрывающиеся списки метода <code>AutoFilter</code>), <code>False</code> (в противном случае)

Пример. Создать таблицу «Социальная стипендия» (рис. 8.8). Таблица состоит из следующих полей: фамилия студента, курс, группа, средняя зарплата отца, средняя зарплата матери, число членов семьи, доход на одного члена семьи, социальная стипендия. С помощью пользовательской формы создать диалоговое окно для ввода исходных данных (рис. 8.9). Социальная стипендия начисляется, если доходы семьи студента, ниже прожиточного минимума. Отсортировать таблицу по курсам, группам и фамилиям. С помощью метода `AutoFilter` создать список студентов, которым начислена стипендия.

	A	B	C	D	E	F	G	H	I
1	Социальная стипендия								
2									
3	№	Фамилия	Курс	Группа	Сред. З/п отца	Сред. З/п матери	Число членов семьи	Доход на 1 члена семьи	Социальная стипендия
4	11	Ахтареева	3	991	10800	8300	3	6366,67	0
5	20	Курбонов	3	991	15700	8900	4	6150,00	0
6	21	Митрофанова	3	991	7400	5600	4	3250,00	1600
7	28	Мосолов	3	991	8300	5400	4	3425,00	1600
8	13	Мухутдинова	3	991	15600	7500	4	5775,00	0
9	14	Рахметов	3	991	18700	6900	4	6400,00	0
10	10	Салахиев	3	991	15600	10800	4	6600,00	0
11	24	Сафиуллин	3	991	15200	7600	4	5700,00	0
12	15	Шайхулисламов	3	991	15700	10300	4	6500,00	0
13	22	Шакирова	3	991	9600	6300	4	3975,00	1600

Рис. 8.8.

Допустим, прожиточный минимум равен 4500 рублей, а размер социальной стипендии - 1600 рублей. Эти значения запишем в ячейки L2 и L3. Ячейке L2 присвоим имя *min*, а L3 - имя *st*.

Рис. 8.9.

Отсортируем таблицу по трем критериям: курсам, группам и фамилиям.

Определим число заполненных строк и выделим диапазон:

```
nom = Cells(1, 1).CurrentRegion.Rows.Count + 1
```

```
Range("A3:I" + Trim(Str(nom))).Select
```

В методе `Sort` за выбор поля, по которому производится первоначальная сортировка, отвечает аргумент `Key1`. В данном случае

для выбора поля Курс аргументу Key1 присвоим значение Range("C4"). Порядок сортировки по первому критерию устанавливается аргументом Order1. Сортируем по возрастанию, поэтому аргументу Order1 присваиваем xlAscending. Вторичная сортировка происходит по полю Группа по возрастанию, аргументам Key2 и Order2 присваиваем Range("D4") и xlAscending соответственно. Сортировка по третьему критерию происходит по полю Фамилия по возрастанию, аргументам Key3 и Order3 присваиваем Range("B4") и xlAscending соответственно. Аргумент Header отвечает за идентификацию полей, в данном случае идентификатором выбрана первая строка диапазона, содержащая заголовки, эта строка не сортируется, аргументу присваиваем значение xlYes. При сортировке регистры учитываться не будут, аргументу MatchCase присвоим значение False. Аргумент Orientation отвечает за направление сортировки, в нашем примере сортировка осуществляется по строкам, аргументу присваиваем значение xlTopToBottom. Таким образом имеем:

```
Range("A3:I" + Trim(Str(nom))).Select  
Selection.Sort Key1:=Range("C4"), Order1:=xlAscending, _  
Key2:=Range("D4"), Order2:=xlAscending, Key3:=Range("B4"), _  
Order3:=xlAscending, Header:=xlYes, OrderCustom:=1, _  
MatchCase:=False, Orientation:=xlTopToBottom
```

Чтобы оставить в таблице только тех студентов, которым начислена стипендия, применим метод AutoFilter. Выделим диапазон "A3:I3", содержащий заголовки полей таблицы. К этому диапазону применим метод AutoFilter. Отфильтруем в таблице данные о студентах, которым начислена социальная стипендия. В методе AutoFilter за выбор поля, в котором производится фильтрация, отвечает аргумент field. Аргументу field присвоим значение 9 («Социальная стипендия» по счету девятое поле). За критерии, по которым производится фильтрация, отвечают аргументы criteria1 и criteria2. В нашем примере фильтрация производится по одному критерию - социальная стипендия, поэтому аргументу criteria1 присвоим значение ">0", а аргументу operator – значение xlAnd.

Таким образом, имеем:

```
Range("A3:I3").Select  
Selection.AutoFilter  
Selection.AutoFilter field:=9, Criteria1:=">0", Operator:=xlAnd
```

	A	B	C	D	E	F	G	H	I
1	Социальная стипендия								
2									
3	№	Фамилия	Курс	Группа	Сред. З/п отца	Сред. З/п матер	Число членов семьи	Доход на 1 члена семьи	Социальная стипендия
6	21	Митрофанова	3	991	7400	5600	4	3250,00	1600
7	28	Мосолов	3	991	8300	5400	4	3425,00	1600
13	22	Шакирова	3	991	9600	6300	4	3975,00	1600
18	18	Давлетшина	5	971	9300	7200	4	4125,00	1600
23	21	Калимуллина	5	971	10400	6400	4	4200,00	1600
26	26	Миргазизова	5	971	9700	6200	4	3975,00	1600

Рис. 8.10.

Чтобы отменить фильтр запишем следующую инструкцию:
`Range("A3:I3").AutoFilter`

Порядок выполнения программы:

Процедура `userform_Initialize()` активизируется всякий раз, когда форма загружается в память посредством выполнения оператора `Load` или с помощью метода `Show`. Считывает размеров прожиточного минимума и социальной стипендии в переменные. Заполняет данными поля со списком для ввода номеров курсов и групп.

Нажатие кнопки «Добавить строку» запускает на выполнение процедуру `CommandButton1_Click()`. Эта процедура определяет номер первой пустой строки рабочего листа, считывает данные из диалогового окна и записывает их в первую пустую строку таблицы. Определяет доход на одного члена семьи и начисляет социальную стипендию.

Нажатие кнопки «Сортировать» запускает на выполнение процедуру `CommandButton3_Click()`. Эта процедура производит сортировку таблицы по трем критериям: курсам, группам и фамилиям.

Нажатие кнопки «Применить автофильтр» запускает на выполнение процедуру `CommandButton4_Click()`. Эта процедура производит фильтрацию в таблице данных о студентах, которым начислена социальная стипендия.

Нажатие кнопки «Отменить автофильтр» запускает на выполнение процедуру `CommandButton5_Click()`. Эта процедура отменяет автофильтр.

Нажатие кнопки «Заккрыть» запускает на выполнение процедуру CommandButton2_Click(). Эта процедура закрывает диалоговое окно.

Нажатие кнопки «Очистить» запускает на выполнение процедуру CommandButton6_Click(). Эта процедура удаляет введенные в таблицу данные.

Код модуля формы

```
Dim nomer, st, min As Integer
```

```
Private Sub userform_Initialize()
```

```
' Считывание размеров прожиточного минимума и  
' социальной стипендии в переменные
```

```
st = Range("L3").Value
```

```
min = Range("L2").Value
```

```
' Заполнение полей со списком
```

```
With ComboBox1
```

```
.List = Array(1, 2, 3, 4, 5)
```

```
.ListIndex = -1
```

```
End With
```

```
With ComboBox2
```

```
.List = Array(971,981, 991, 951)
```

```
.ListIndex = -1
```

```
End With
```

```
End Sub
```

```
Private Sub CommandButton1_Click()
```

```
' Процедура считывания информации из диалогового окна,
```

```
' записи ее в таблицу, определения дохода на одного
```

```
' члена семьи и начисления социальной стипендии
```

```
' nomer - номер первой пустой строки рабочего листа
```

```
nomer = Cells(1, 1).CurrentRegion.Rows.Count + 1
```

```
' Считывание информации из диалогового окна
```

```
Range("A" + Trim(Str(nomer))).Value = nomer - 3
```

```
Range("B" + Trim(Str(nomer))).Value = TextBox1.Text
```

```
If ComboBox1.ListIndex <> -1 Then
```

```
Range("C" + Trim(Str(nomer))).Value = _
```

```
ComboBox1.List(ComboBox1.ListIndex, 0)
```

```
Else
```

```

MsgBox "Номер курса выберите из списка", 48
Exit Sub
End If
Range("D" + Trim(Str(nomer))).Value = ComboBox2.Text Range("E" +
Trim(Str(nomer))).Value = TextBox4.Text
Range("F" + Trim(Str(nomer))).Value = TextBox5.Text
Range("G" + Trim(Str(nomer))).Value = TextBox6.Text
' Очистка полей формы для ввода очередной записи
TextBox1.Text = ""
TextBox4.Text = ""
TextBox5.Text = ""
TextBox6.Text = ""
ComboBox1.ListIndex = -1
ComboBox2.ListIndex = -1
Range("L5").Value = nomer
' Запись формулы для вычисления дохода на одного члена семьи
Cells(nomer, 8).Activate
ActiveCell.FormulaR1C1 = "=Round((RC[-3]+RC[-2])/RC[-1],2)"
' Запись формулы для начисления социальной стипендии
Cells(nomer, 9).Activate
ActiveCell.FormulaR1C1 = "=IF(RC[-1]<=min,st,0)"
End Sub

Private Sub CommandButton3_Click()
' Процедура сортировки таблицы по трем критериям:
' курсам и группам и фамилиям
Dim nom As Integer
nom = Cells(1, 1).CurrentRegion.Rows.Count
Range("A3:I" + Trim(Str(nom))).Select
Selection.Sort Key1:=Range("C4"), Order1:=xlAscending, _
Key2:=Range("D4"), Order2:=xlAscending, Key3:=Range("B4"), _
Order3:=xlAscending, Header:=xlYes, OrderCustom:=1, _
MatchCase:=False, Orientation:=xlTopToBottom
End Sub
'

Private Sub CommandButton4_Click()
' Фильтрация в таблице данных о студентах,
' которым начислена социальная стипендия

```

```
Range("A3:I3").Select
Selection.AutoFilter
Selection.AutoFilter field:=9, Criteria1:=">0", Operator:=xlAnd
End Sub
```

```
Private Sub CommandButton2_Click()
' Процедура закрытия диалогового окна
Hide
End Sub
```

```
Private Sub CommandButton5_Click()
' Процедура отмены автофильтра
Range("A3:I3").AutoFilter
End Sub
```

```
Private Sub CommandButton6_Click()
nomer = Cells(1, 1).CurrentRegion.Rows.Count
Range("A4:I" + Trim(Str(nomer))).Clear
End Sub
```

Задания для самостоятельного выполнения:

1. Создать программу приближенного решения следующих уравнений методом GoalSeek, результаты вычислений вывести на рабочий лист Excel:

- a) $ax^3 + 3b\sin x = 0$
- b) $5ax^{1/3} + 3btg4x = 0$
- c) $\ln(ax^3) + 3b\cos(e^x) = 0$

2. Создать программу заполнения таблицы значений функций, с использованием метода AutoFill:

$$a) \quad F(x) = \begin{cases} -x^2 + 3x + 9, & \text{если } x \geq 3 \\ \frac{1}{x^3 - 6}, & \text{если } x < 3 \end{cases} \quad \text{на отрезке } [-5; 5] \text{ с}$$

шагом 0,4.

$$b) \quad F(x) = \begin{cases} 9, & \text{если } x \leq -3 \\ \frac{1}{x^2 + 1}, & \text{если } x > -3 \end{cases} \quad \text{на отрезке } [-10; 10] \text{ с}$$

шагом 1.

$$c) \quad F(x) = \begin{cases} 3x - 9, & \text{если } x \leq 7 \\ \frac{1}{x^2 - 4}, & \text{если } x > 7 \end{cases} \quad \text{на отрезке } [-7; 17] \text{ с}$$

шагом 1.

3. Построить таблицу, подсчитывающую заработную плату сотрудников. Таблица состоит из следующих полей: фамилия сотрудника, номер подразделения, оклад, профсоюзный взнос, подоходный налог, сумма к выдаче. С помощью пользовательской формы создать диалоговое окно для ввода исходных данных. Подоходный налог взимается в размере 13% от оклада, профсоюзный взнос – 1%. Отсортировать таблицу по подразделениям и фамилиям. С помощью метода AutoFilter создать список сотрудников:

- a) данного подразделения;
- b) имеющих оклад больше 30000 рублей.

4. Создать таблицу «Абитуриент». Таблица состоит из следующих полей: фамилия абитуриента и баллы по ЕГЭ по математике, информатике и иностранному языку, общая сумма баллов, отметка о зачислении. С помощью пользовательской формы создать диалоговое окно для ввода исходных данных. Вычислить общую сумму баллов и сделать отметку о зачислении абитуриента в ВУЗ. Если сумма баллов больше или равна проходному и баллы по информатике не ниже 75, абитуриент зачисляется. Отсортировать таблицу по общему количеству баллов и фамилиям. С помощью метода AutoFilter создать список абитуриентов:

- a) зачисленных в ВУЗ;
- b) имеющих балл по информатике не ниже 75.

9. Создание макросов

Рассмотрим, как создается макрос на языке VBA с использованием MacroRecorder.


MacroRecorder — представляет собой транслятор, создающий программу (макрос) на языке VBA, которая является результатом перевода на язык VBA действий пользователя с момента запуска макрорекордера до окончания записи макроса. Основное назначение макросов — автоматизация работы пользователя. Кроме этого, созданный код макроса может служить основой для дальнейших разработок.

При записи макроса запоминаются все действия пользователя, будь то нажатие клавиши или выбор определенной команды меню, которые автоматически преобразуются в программный код на языке VBA.

Каждому макросу дается имя, а для быстрого запуска макроса можно создать или присвоить ему “горячую” клавишу (клавишу, по нажатию на которую будет производиться запуск макроса). После запуска макрос будет автоматически выполнен тем приложением, в котором он создан и запущен. При выполнении макроса компьютер воспроизведет все действия пользователя.

Макрос — это именованная последовательность заданных пользователем команд и действий, хранящаяся в форме программы на языке VBA.

Для записи макроса с помощью макрорекордера необходимо:

1. Перейти на вкладку **Разработчик**, в группе **Код** нажать на кнопку **Запись макроса** .

2. В открывшемся диалоговом окне **Запись макроса** (рис. 9.1) установить параметры записываемой процедуры (ее имя, описание, сочетание клавиш для выполнения записанной процедуры) и нажать на кнопку **ОК**. На экране на вкладке **Разработчик** в группе **Код** кнопка **Запись макроса** изменится на кнопку **Остановить запись** (рис. 9.2).

3. Выполнить действия, которые нужно записать.

4. Нажать кнопку **Остановить запись** на вкладке **Разработчик** в группе **Код**.

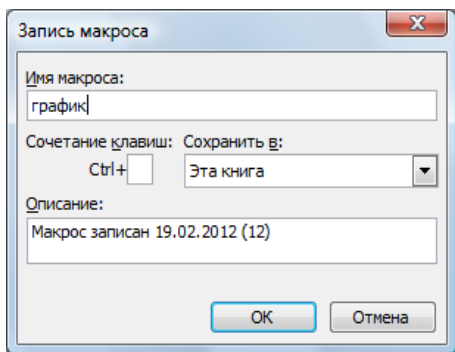


Рис. 9.1. Окно **Запись макроса**

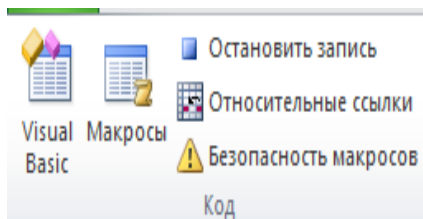


Рис. 9.2. Группа **Код** на вкладке **Разработчик** в режиме записи макроса

Для просмотра и редактирования созданной процедуры:

1. Нажать кнопку **Макросы** в группе **Код** на вкладке **Разработчик**, затем в открывшемся диалоговом окне **Макрос** (рис. 9.3) выбрать в списке имя нужного макроса и нажать кнопку **Изменить**, далее откроется главное окно редактора Microsoft Visual Basic и окно **Модуль (Module)** с текстом выбранного макроса.

2. Внести в текст макроса необходимые изменения и закрыть окно редактора.

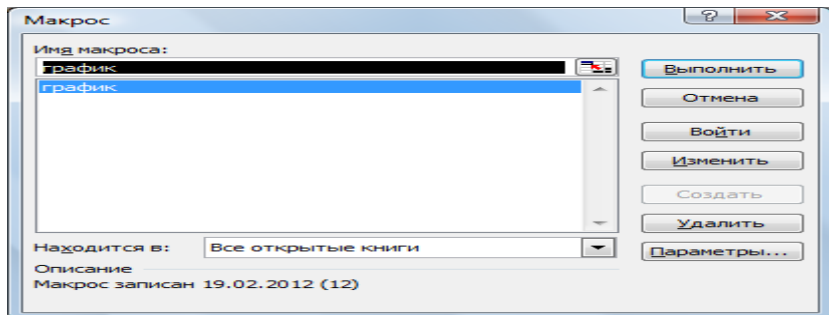


Рис. 9.3. Диалоговое окно **Макрос**

Для выполнения макроса:

Нажать кнопку **Макросы** в группе **Код** на вкладке **Разработчик**, затем в открывшемся диалоговом окне **Макрос** выбрать в списке имя нужного макроса и нажать кнопку **Выполнить**.

Пример. Создать макрос построения графиков функций: $y_1 = |\sin x| + |\cos x|$ и $y_2 = 3 \sin \sqrt{x} + 0,35x - 1,8$ на отрезке $[-5;5]$ с шагом $h=0,5$.

Для создания макроса необходимо выполнить следующие действия:

1. Открыть новую книгу. Книгу сохранить с расширением **.xlsm**, при сохранении документа в поле со списком **Тип файла** выбрать **Книга Excel с поддержкой макросов**.
2. Создать таблицу по приведенному образцу (рис. 9.4).
3. Выделить диапазон ячеек **A1:C22**. Перейти на вкладку **Разработчик**, в группе **Код** нажать на кнопку **Запись макроса**. Появится диалоговое окно **Запись макроса** (рис.9.1). Ввести название **Графики** и нажать на кнопку **ОК**.
4. Перейти на вкладку **Вставка** ленты. В группе **Диаграммы** выбрать тип графика «Точечная» и вид графика. Перейти на

вкладку **Макет**, в группе **Подписи** нажать кнопку **Название диаграммы** и в открывшемся окне выбрать **Над диаграммой**, в области диаграммы появится поле **Название диаграммы**, где ввести **Графики функций**.

- Изменить параметры осей диаграммы. На вкладке **Макет** в группе **Оси** открыть список **Оси**, затем открыть список **Основная горизонтальная ось**, из открывшегося списка выбрать **Дополнительные параметры основной горизонтальной оси** Откроется окно **Формат оси**, где в области **Параметры оси** изменить параметры по образцу, приведенному на рисунке 9.5. Параметры основной вертикальной оси изменить по образцу, приведенному на рисунке 9.6.
- На вкладке **Разработчик** в группе **Код** нажать кнопку **Остановить запись**.

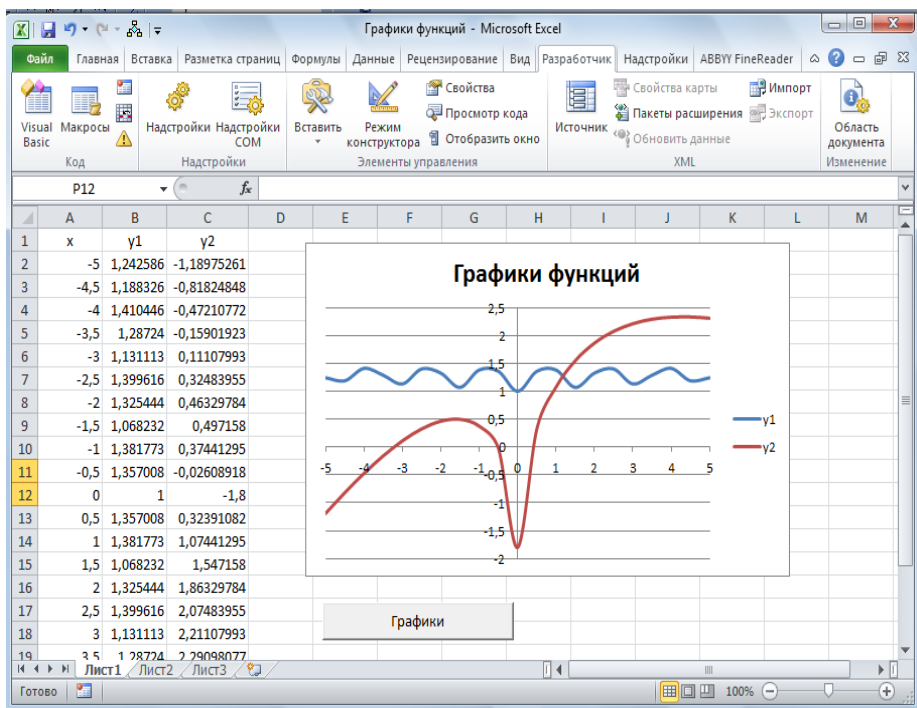


Рис. 9.4.

Параметры оси

минимальное значение: авто фиксированное -5,0

максимальное значение: авто фиксированное 5,0

цена основных делений: авто фиксированное 1,0

цена промежуточных делений: авто фиксированное 0,5

обратный порядок значений

догарифмическая шкала Основная: 10

Цена деления: нет

Отображать на диаграмме

Рис. 9.5. Параметры основной горизонтальной оси

Параметры оси

минимальное значение: авто фиксированное -2,0

максимальное значение: авто фиксированное 2,5

цена основных делений: авто фиксированное 0,5

цена промежуточных делений: авто фиксированное 0,1

обратный порядок значений

догарифмическая шкала Основная: 10

Цена деления: нет

Отображать на диаграмме

Рис. 9.6. Параметры основной вертикальной оси

Назначение макроса командной кнопке

Запустить макрос на выполнение можно с помощью командной кнопки. Вставить командную кнопку непосредственно на рабочий лист Excel можно с помощью панели «Элементы управления». Затем необходимо назначить ей наш макрос. Щелкнуть правой кнопкой мыши на кнопке и в открывшемся контекстном меню выбрать команду «Свойства». Откроется окно «Properties», где изменить надпись на кнопке. Еще раз щелкнуть правой кнопкой мыши на кнопке и в открывшемся контекстном меню выбрать команду «Исходный текст». Откроется окно кода рабочего листа, где после заголовка процедуры, ввести инструкцию: Call Графики.

```
Private Sub CommandButton1_Click()
    Call Графики
End Sub
```

Чтобы запустить макрос достаточно будет нажать кнопку «Графики» (рис. 9.4).

Назначение макроса кнопке на панели быстрого доступа

Макрос Графики можно назначить кнопке на панели быстрого доступа.

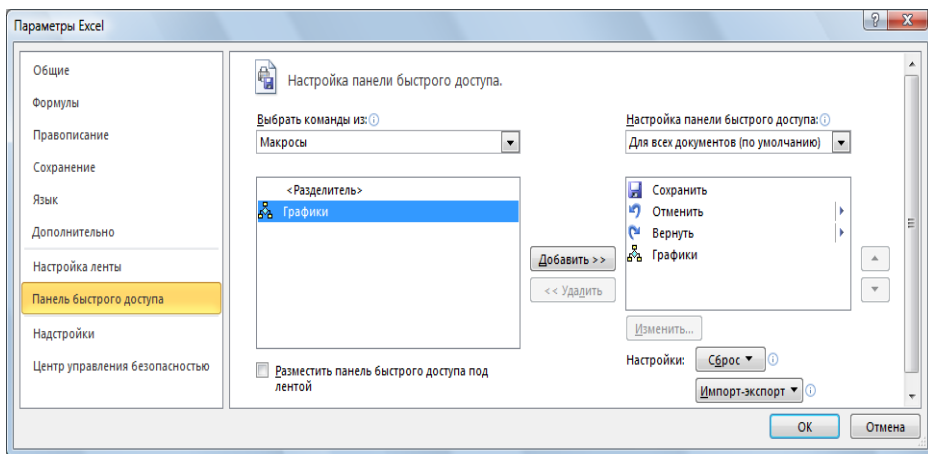


Рис. 9.7. Выбор объекта **Макросы** в окне **Параметры Excel**

1. На панели быстрого доступа открыть список **Настройка панели быстрого доступа** и выбрать команду **Другие команды**.
2. В открывшемся окне **Параметры Excel** категории **Панель быстрого доступа** выбрать в поле со списком **Выбрать команды из** объект **Макросы** (рис.9.7).
3. В левом столбце выбрать макрос **Графики** и с помощью кнопки **Добавить >>** перенести его в правый столбец. В правом столбце становится активным кнопка **Изменить**.
4. В открывшемся окне **Изменение кнопки** при необходимости в поле **Отображаемое имя** можно изменить имя для макроса, которое будет всплывающей подсказкой на панели быстрого доступа.

Параметры макросов

На вкладке **Разработчик** в группе **Код** находится кнопка **Безопасность макросов**, которая открывает окно **Центр управления безопасностью** в категории **Параметры макросов** (рис. 9.8). В этом окне можно выбрать требуемый параметр, чтобы предотвратить нежелательное выполнение вредоносного кода, который может содержаться в макросах, полученных из неизвестных источников.

Выбрав слева в окне **Центр управления безопасностью** категорию **Надежные расположения** и нажав справа кнопку **Добавить новое расположение**, нужно указать место, откуда файлы, содержащие код VBA, будут открываться без блокирования соответствующих действий (рис. 9.9).

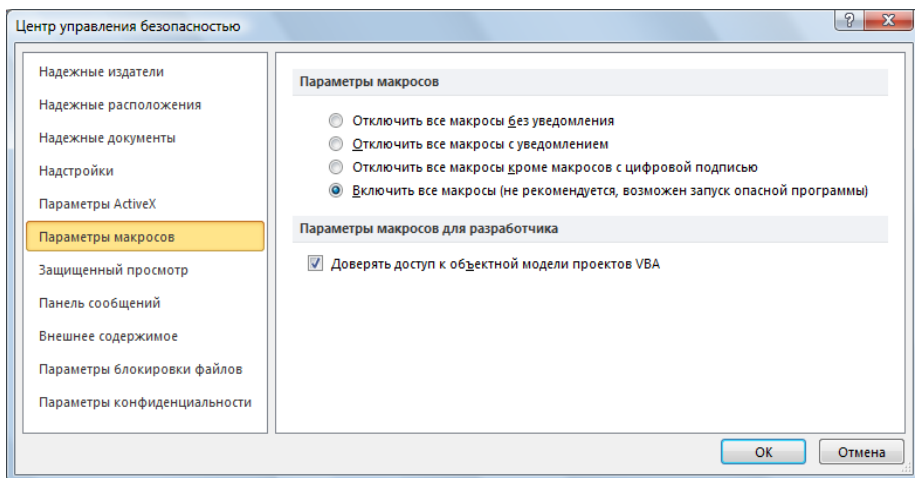


Рис. 9.8. Окно **Центр управления безопасностью**, категория **Параметры макросов**

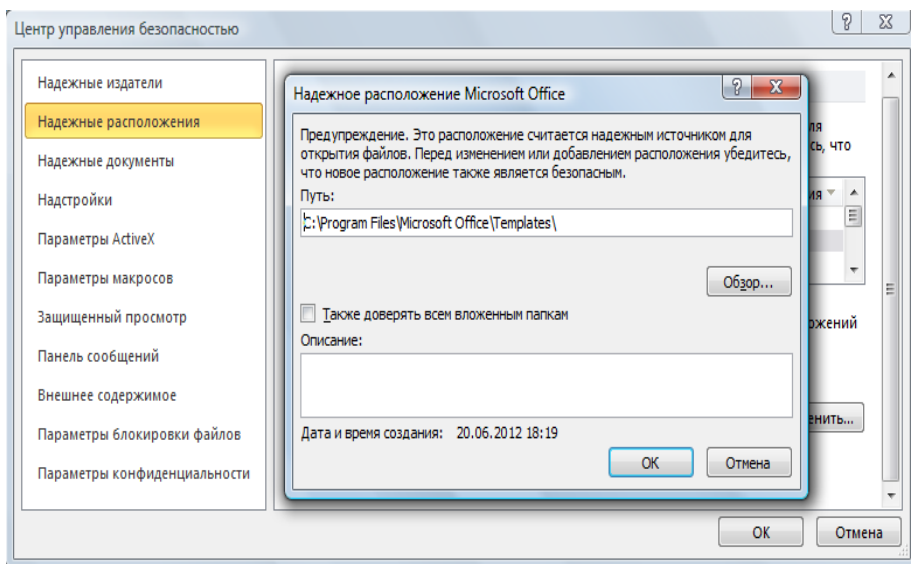


Рис. 9.9. Окно **Центр управления безопасностью**, категория **Надежные расположения**

Примечание. Параметры макросов необходимо установить в самом начале работы с VBA.

Задания для выполнения:

1. Клиент разместил в банке 100000 рублей под 5% годовых. Начисление процентов производится один раз в год. Определить размера вклада и дохода через N лет. Использовать финансовую функцию БС. Создать макрос построения диаграммы суммы вклада и дохода.

- Ввести исходные данные (рис. 9.10).
- В ячейку **B4** ввести формулу определения размера вклада через указанное количество лет.
- В ячейку **B5** ввести формулу вычисления суммы дохода.
- Выделить диапазон **A1:B1**, затем выделить диапазон **A5:B5** при нажатой клавише **Ctrl**.
- Создать макрос построения диаграммы суммы вклада и дохода. Тип диаграммы выбрать Гистограмма.
- Для вызова макроса задать сочетание клавиш **Ctrl+q**.

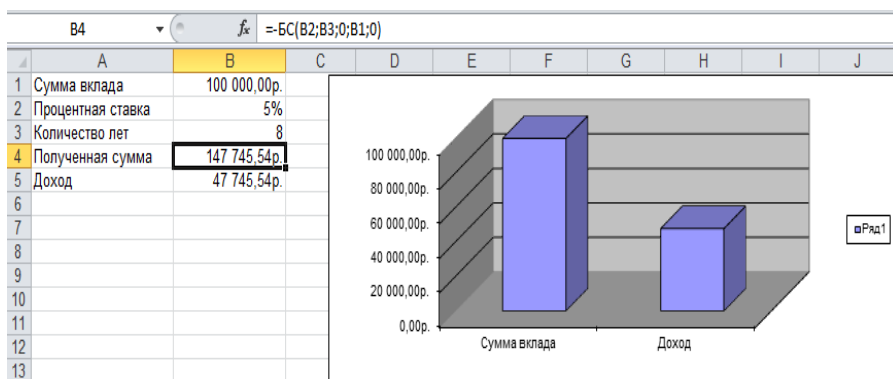


Рис. 9.10.

2. Создать макрос построения графиков функций: $y_1 = \frac{e^{\sqrt{x^2+1}}}{10}$ и

$y_2 = \cos \sqrt{2x^2+1}$ на отрезке $[-3;3]$ с шагом $h=0,5$. Макросу назначить кнопку на панели быстрого доступа.

3. Создать таблицу «Список группы» (рис. 9.11). Таблица состоит из следующих полей: Фамилия, Имя, Отчество, Год рождения, Телефон. Заполнить таблицу данными.

- 1) Создать макрос сортировки таблицы по фамилиям и именам.

- 2) Создать макрос сортировки таблицы по годам рождения и фамилиям.
- 3) Создать макрос изменения цвета и шрифта в диапазоне:
 - A1–F1 — полужирный шрифт, кегль 16, зеленый фон, буквы белые;
 - A2–A11 — розовый фон;
 - B2–B11 — голубой фон;
 - C2–C11 — серый фон;
 - D2–D11 — розовый фон;
 - E2–E11 — голубой фон;
 - F2–F11 — серый фон.
- 4) Создать макрос возврата таблицы в исходное состояние.
- 5) Созданным макросам назначить командные кнопки.

	A	B	C	D	E	F
1	№	Фамилия	Имя	Отчество	Год рождения	Телефон
2	1	Алексеева	Ольга	Викторовна	1996	267-43-15
3	2	Петров	Олег	Андреевич	1995	512-18-67
4	3	Зайцева	Ирина	Васильевна	1994	236-45-35
5	4	Яковлев	Сергей	Анатольевич	1995	269-45-32
6	5	Симонова	Алина	Петровна	1994	522-56-85
7	6	Мусин	Андрей	Ахатович	1993	89275634931
8	7	Иванова	Анна	Юрьевна	1996	89270084567
9	8	Аюпова	Амина	Салиховна	1994	89175510856
10	9	Ильин	Георгий	Владимирович	1995	89169354002
11	10	Васин	Петр	Георгиевич	1996	89600547258

Рис. 9.11.

10. Компьютерное моделирование физических задач средствами VBA в Excel

Раздел «Компьютерные модели в физике» занимает в курсе компьютерного моделирования значительный объем.

Рассмотрим компьютерное моделирование средствами VBA в Excel следующих физических задач:

- свободное падение тела с учетом сопротивления среды;
- движение тела, брошенного под углом к горизонту, с учетом сопротивления среды.

В рассматриваемых физических задачах фундаментальную роль играет второй закон Ньютона. Он гласит, что ускорение, с которым движется тело, прямо пропорционально действующей на него силе (если их несколько, то равнодействующей, т.е. векторной сумме сил) и обратно пропорционально его массе: $\vec{a} = \frac{\vec{F}}{m}$.

Если сила или масса не являются величинами постоянными, необходимо записать этот закон в более общей математической форме. Поскольку ускорение есть приращение скорости $\vec{v}(t)$, а скорость — приращение перемещения $\vec{s}(t)$, то

$$\frac{d\vec{s}}{dt} = \vec{v}(t), \quad \frac{d\vec{v}}{dt} = \frac{\vec{F}(t, \vec{v}, \vec{s})}{m(t, \vec{v}, \vec{s})} \quad (1)$$

10.1. Задача о свободном падении тела с учетом сопротивления среды

Рассмотрим следующую задачу[6]: парашютист совершает затяжной прыжок с высоты 2000 м. Считая массу парашютиста заданной ($m=80$ кг), определить, начиная с какого времени, после начала полета его скорость становится постоянной. Построить графики зависимости скорости и высоты падения от времени. Решить задачу методами Рунге-Кутты и Эйлера-Коши. Для решения задачи создать диалоговое окно с помощью формы (рис. 10.1).

Решение.

Математическая модель свободного падения тела следует из уравнения второго закона Ньютона с учетом двух сил, действующих на падающее тело – силы тяжести и силы сопротивления среды:

$$\vec{F} = \vec{F}_{\text{сопр}} + \vec{F}_{\text{тяж}}, \quad \text{где } \vec{F}_{\text{сопр}} = k_1 \vec{v} + k_2 \vec{v}^2, \quad \vec{F}_{\text{тяж}} = m \vec{g}.$$

Рассматриваемое движение является одномерным. Проецируя силу \vec{F} , скорость \vec{v} и перемещение \vec{h} на ось, направленную вертикально вниз, из (1) получаем

$$\frac{dh}{dt} = v, \tag{2}$$

$$\frac{dv}{dt} = \frac{mg - k_1v - k_2v^2}{m}$$

В конкретных задачах можно одной из составляющих силы сопротивления пренебречь, если она заведомо много меньше другой. Скорость движения при затяжном прыжке достаточно большая, поэтому вкладом линейной составляющей силы сопротивления k_1v можно пренебречь. Тогда из (2) получим систему дифференциальных уравнений:

$$\frac{dh}{dt} = v, \tag{3}$$

$$\frac{dv}{dt} = \frac{mg - k_2v^2}{m}$$

Здесь v – скорость, t – время, h – высота, m – масса, g – ускорение свободного падения, k_2 – коэффициент квадратичной составляющей скорости.

Входными параметрами модели являются: начальная высота тела; начальная скорость тела; величины, определяющие коэффициенты сопротивления среды. В рассматриваемой задаче начальная высота – 2000м, начальная скорость – $v_0=0$.






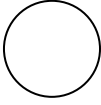

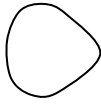
Величина коэффициента k_2 пропорциональна площади сечения тела S , поперечного по отношению к потоку, и плотности среды $\rho_{\text{среды}}$, а также зависит от формы тела. Обычно представляют $k_2=0,5 \cdot c \cdot S \cdot \rho_{\text{среды}}$, где c – коэффициент лобового сопротивления. c – безразмерная величина.

Выберем значение c как среднее между коэффициентами лобового сопротивления диска и полусферы $c=1,22$. Плотность воздуха – $\rho_{\text{среды}}=1,29 \text{ кг/м}^3$

Значение площади S поперечного сечения вычислим как произведение роста на полуобхват груди. Если рост (высота тела) =1.7 м, а полуобхват груди (ширина тела) =0.5 м, то $S = 1.7 \cdot 0.5 = 0.85 \text{ м}^2$.

Некоторые значения c приведены в таблице 10.1.

Таблица 10.1 Таблица значений коэффициента лобового сопротивления

		Диск	$c = 1,11$
		Полусфера	$c = 1,33$
		Шар	$c = 0,4$
		Каплевидное тело	$c = 0,045$

Итак, нужно определить характер изменения скорости и высоты со временем, если все параметры, входящие в уравнения системы (3), заданы. При наличии сопротивления, растущего со скоростью, в какой-то момент сила сопротивления сравняется с силой тяжести, после чего скорость больше возрастать не будет. Это будет видно на графике зависимости скорости от времени. Для решения задачи выберем путь численного моделирования, включающий компьютерный эксперимент. При решении будем использовать два численных метода: метод Рунге-Кутты и метод Эйлера-Коши. В модифицированном методе Эйлера – методе Эйлера-Коши предлагается следующий порядок вычислений:

$$\begin{aligned}
 v_{i+1}^* &= v_i + \tau \cdot f(t_i, v_i), \\
 v_{i+1} &= v_i + \tau \cdot \frac{f(t_i, v_i) + f(t_{i+1}, v_{i+1}^*)}{2}.
 \end{aligned}
 \tag{4}$$

Этот метод сначала грубо вычисляет по методу ломаных значение функции $v_{i+1}^* = v_i + \tau \cdot f(t_i, v_i)$, где $f(t_i, v_i) = \frac{mg - k_2 v_i^2}{m}$ – направление интегральной кривой в исходной точке.

Затем вычисляется направление интегральной кривой в новой точке (x_{i+1}, y_{i+1}^*) :

$$f(t_{i+1}, v_{i+1}^*) = \left(mg - k_2 \left(v_i + \tau \frac{mg - k_2 v_i^2}{m} \right)^2 \right) / m$$

В качестве окончательного в (4) берется среднее этих направлений и вычисляется значение функции:

$$v_{i+1} = v_i + \tau \cdot \frac{f(t_i, v_i) + f(t_{i+1}, v_{i+1}^*)}{2}.$$

Таким образом, окончательная формула вычисления скорости методом Эйлера-Коши имеет вид:

$$v_{i+1} = v_i + \frac{\tau}{2} [(mg - k_2 v_i^2) / m + (mg - k_2 \cdot (v_i + \tau (mg - k_2 v_i^2) / m)^2) / m] \quad (5)$$

Методом Рунге-Кутта можно строить схемы различного порядка точности. Наиболее употребительны схемы четвертого порядка:

$$v_{i+1} = v_i + \frac{\tau}{6} \cdot (k_1 + 2k_2 + 2k_3 + k_4),$$

$$k_1 = f(t_i, v_i), \quad k_2 = f\left(t_i + \frac{\tau}{2}, v_i + \frac{\tau}{2} k_1\right), \quad (6)$$

$$k_3 = f\left(t_i + \frac{\tau}{2}, v_i + \frac{\tau}{2} k_2\right), \quad k_4 = f(t_i + \tau, v_i + \tau k_3).$$

При вычислении значения функции v_{i+1} методом Рунге-Кутта производятся четыре оценки производной k_1, k_2, k_3, k_4 и вычисляется средневзвешенная производная. При этом оценки производной на половине шага входят с большим весом, чем ее оценки в начале и в конце шага интегрирования. С помощью средневзвешенной производной вычисляется текущее значение скорости v_{i+1} в конце $i+1$ -го шага интегрирования.

Численное решение задачи методами Рунге-Кутта и Эйлера-Коши с построением графиков получим с помощью программирования средствами VBA в Excel.

Для поставленной задачи создадим VBA-приложение. Для построения графиков зависимостей скоростей v и vk от времени создадим макрос `grafic`. Макрос построения диаграммы создается для конкретного

диапазона данных. Если при создании макроса был выделен диапазон A2:C34, то в макросе для SetSourceData будет установлено:

```
ActiveChart.SetSourceData Source:= Range("Лист1!$A$2:$C$34").
```

При изменении высоты и шага интегрирования, диапазон данных, по которому строится диаграмма, тоже меняется. Например, при $h=3000$ м и $dt=1$ сек. процедура CommandButton1_Click выделит диапазон A2:C94. В макросе построения диаграммы для SetSourceData должен быть установлен соответствующий диапазон. Для того чтобы перестраивать диаграмму, в общей части (General) модуля макроса объявляем глобальную переменную r типа Range:

```
Public r As Range
```

Диапазон, по которому будет строиться диаграмма, устанавливаем в процедуре CommandButton1_Click модуля формы:

```
Set r = Range("A2:C" + Trim(Str(nom))).
```

В макросе для SetSourceData устанавливаем диапазон r:

```
ActiveChart.SetSourceData Source:=r
```

Для размещения диаграммы в диапазоне F5:K19 в макрос добавляем следующие инструкции:

```
With ActiveSheet.ChartObjects(1)
    .Top = Range("F5").Top
    .Left = Range("F5").Left
    .Width = Range("F5:L19").Width
    .Height = Range("F5:L19").Height
End With
```

Созданное приложение можно запустить с листа Excel. Для этого на рабочий лист Excel вставим командную кнопку «Запуск программы». В модуле рабочего листа Лист1 создадим процедуру обработки события нажатия этой кнопки. Для этого нужно активизировать режим конструктора, дважды щелкнуть на данной кнопке. В редакторе VBA реализовать модуль рабочего листа, куда добавить одну инструкцию: UserForm1.Show. Теперь, чтобы отобразить форму достаточно нажать на кнопку «Запуск программы». Вид формы приведен на рисунке 10.1. Диалоговое окно содержит поля ввода данных для вычисления коэффициента k_2 , значение массы тела и шаг интегрирования, а также три командные кнопки для управления работой приложения.

После появления формы нужно ввести исходные данные и нажать на кнопку «Старт». Описание процедуры обработки этого события приведено ниже в коде модуля формы.

масса тела	80
высота тела	1.7
ширина тела	0.5
высота падения	3000
шаг интегрирования	1

Buttons: Старт, Очистить, Заккрыть

Рис.10.1

Для выполнения расчетов в модуле формы организован цикл с предусловием: пока значение высоты h больше нуля. Значение времени в цикле увеличивается на значение приращения dt , в каждом временном слое вычисляются значения высоты и скорости по методам Рунге-Кутта (6) и Эйлера-Коши (5). Для вычисления этих значений методом Рунге-Кутта (6) вызывается функция $gkhv$, которая на данном интервале вычисляет четыре значения приращений для вычисляемой функции, а затем ее текущее значение. Функции $gkhv$ передаются четыре параметра. Если значение первого параметра $p1=1$, то функция возвращает текущее значение высоты h , если $p1=2$, то функция возвращает текущее значение скорости v . Текущее значение скорости vk по методу Эйлера-Коши вычисляется по формуле (5). Вычисленные значения записываются на рабочий лист Excel. После завершения цикла на листе Excel выделяется диапазон с результатами вычислений и вызывается макрос для построения графиков зависимостей скоростей v и vk от времени. На рабочем листе Лист1 появляются графики функций. Для ознакомления с результатами вычислений форму нужно закрыть, нажав на кнопку «Заккрыть». Для выполнения приложения с другими исходными данными, нажать на рабочем листе на кнопку «Запуск программы» и для удаления результатов предыдущих вычислений на появившейся форме нажать на кнопку «Очистить». Для завершения работы с приложением нажать на кнопку «Заккрыть».

Код модуля формы

Const $g = 9.8$, $c = 1.22$, $ro = 1.29$

' v – скорость по методу Рунге-Кутта, vk – скорость по методу Эйлера-Коши

Dim h, v, v0, vk As Double

Dim rost, gr, k2, m, t, dt, a, x As Single

Dim nom As Integer

' Функция вычисления значения скорости

Function fv(hn, vn As Variant) As Double

$fv = g - (k2 * vn ^ 2) / m$

End Function

' Функция вычисления значения высоты

Function fh(hn, vn As Variant) As Double

$fh = vn$

End Function

' Функция вычисления текущих значений h и v методом Рунге-Кутта,

' если параметр p1=1, функция возвращает текущее значение h

' если p1=2, функция возвращает текущее значение v

Function rkhv(p1, dt, hn, vn As Variant) As Double

Dim D(3) As Double

Dim dD(4) As Double

Dim dsr As Double

Select Case p1

Case 1

$dD(1) = fh(hn, vn)$

$D(1) = hn - dD(1) * dt / 2$

$dD(2) = fh(D(1), vn)$

$D(2) = hn - dD(2) * dt / 2$

$dD(3) = fh(D(2), vn)$

$D(3) = hn - dD(3) * dt$

$dD(4) = fh(D(3), vn)$

$dsr = (1 / 6) * (dD(1) + 2 * dD(2) + 2 * dD(3) + dD(4))$

$rkhv = hn - dsr * dt$

Case 2

$dD(1) = fv(hn, vn)$

$D(1) = vn + dD(1) * dt / 2$

$dD(2) = fv(hn, D(1))$

$D(2) = vn + dD(2) * dt / 2$

$dD(3) = fv(hn, D(2))$

$D(3) = v_n + dD(3) * dt$

$dD(4) = f_v(h_n, D(3))$

$dsr = (1 / 6) * (dD(1) + 2 * dD(2) + 2 * dD(3) + dD(4))$

$rkhv = v_n + dsr * dt$

End Select

End Function

' Процедура обработки события Click для CommandButton1
' (кнопка «Старт»)

Private Sub CommandButton1_Click()

m = Val(TextBox1.Text)

rost = Val(TextBox2.Text)

gr = Val(TextBox3.Text)

h = Val(TextBox4.Text)

dt = Val(TextBox5.Text)

$k2 = 0.5 * ro * st * gr * c * ro$

$a = k2 / m$

t = 0 : v0 = 0 : v = v0

vk = 0 : nom = 3

' Вывод начальных значений t, h, v на рабочий лист

Range("A3").Value = t

Range("B3").Value = v

Range("C3").Value = vk

Range("D3").Value = h

Do While h > 0 ' начало цикла расчетов пока h > 0

t = t + dt

h = rkhv(1, dt, h, v) ' вычисление текущего значения h

' Вычисление скорости по методу Рунге-Кутты

v = rkhv(2, dt, h, v) ' вычисление текущего значения v

' Вычисление скорости по методу Эйлера-Коши

$x = (m * g - k2 * v ^ 2) / m$

$vk = vk + dt / 2 * (x + (m * g - k2 * (v + dt * x) ^ 2) / m)$

'вычисление текущего значения vk

nom = nom + 1

' Вывод результатов вычислений на рабочий лист

Range("A" + Trim(Str(nom))).Value = t

Range("B" + Trim(Str(nom))).Value = v

Range("C" + Trim(Str(nom))).Value = vk

```

Range("D" + Trim(Str(nom))).Value = h
Loop
    ' Выделение на рабочем листе диапазона с результатами
Range("J3").Value = nom - 3
Set r = Range("A2:C" + Trim(Str(nom)))
r.Select
' Вызов макроса для построения графика зависимости скорости
' от времени
Call grafic
' Вывод числа итераций цикла
Range("J3").Value = nom - 3
End Sub

' Процедура обработки события Click для CommandButton2
' (кнопка «Очистить»)
Nom=Cells(3,1).Current Region.Rows.Count
Range(Cells(3,1), Cells(nom,4)). Clear
Range("J2").Clear
Range("J3").Clear
' Удаление с активнорбочего листа всех графических объектов
If ActiveSheet.ChartObjects.Count > 0 Then
ActiveSheet.ChartObjects.Delete
End If
TextBox4.Text = ""
TextBox5.Text = ""
End Sub

' Процедура обработки события Click для CommandButton3
' (кнопка «Закреть»)
Private Sub CommandButton3_Click()
UserForm1.Hide
End Sub

```

Результаты исследования задачи методами Рунге-Кутта и Эйлера-Коши представлены на рис.10.2.

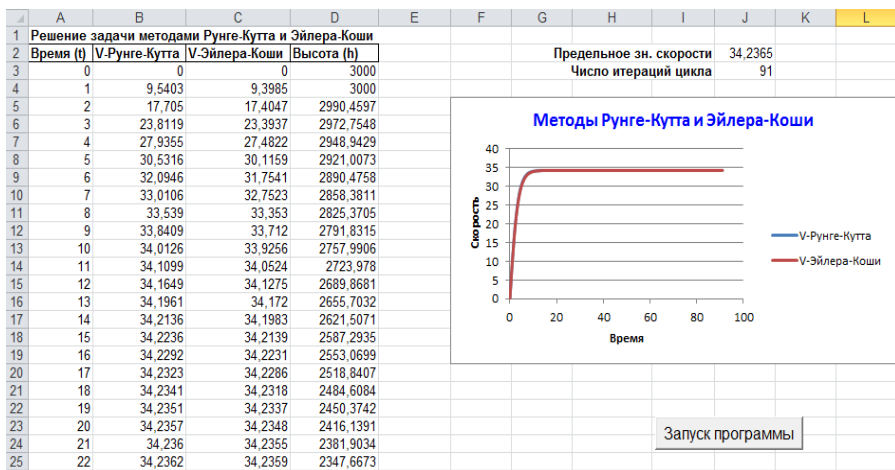


Рис. 10.2.

Задания для самостоятельного выполнения

Задание №1

Парашютист совершает затяжной прыжок с высоты 2000 м. Считая массу, рост, полуобхват грудной клетки заданными (данные приведены в таблице), определить, начиная с какого времени, после начала полета скорость «безпарашютиста» становится постоянной. Построить графики зависимости скорости падения от времени и высоты полета от времени. В последнем столбце таблицы указан метод решения задачи.

№ варианта	Масса (кг)	Рост (м)	Полуобхват грудной клетки (м)	Метод решения
1	60	1,75	0,45	Рунге-Кутта
2	50	1,40	0,3	Эйлера-Коши
3	80	1,65	0,33	Рунге-Кутта
4	65	1,75	0,55	Эйлера-Коши
5	80	1,8	0,45	Рунге-Кутта
6	70	1,65	0,4	Эйлера-Коши
7	60	1,7	0,45	Рунге-Кутта
8	64	1,53	0,4	Эйлера-Коши
9	65	1,8	0,5	Рунге-Кутта
10	80	1,62	0,37	Эйлера-Коши
11	50	1,65	0,35	Рунге-Кутта

12	60	1,75	0,45	Эйлера-Коши
13	67	1,72	0,32	Рунге-Кутта
14	60	1,65	0,35	Эйлера-Коши
15	54	1,85	0,5	Рунге-Кутта

Задание №2

При решении всех вариантов использовать метод Рунге-Кутта.

1. Парашютист прыгает и сразу открывает парашют. В какой момент скорость парашютиста станет постоянной, если радиус парашюта $R = 3$ м, масса парашютиста $m = 90$ кг.
2. Парашютист прыгает и сразу открывает парашют. В какой момент скорость парашютиста станет постоянной, если радиус парашюта $R = 2,5$ м, масса парашютиста $m = 80$ кг.
3. Парашютист прыгает и сразу открывает парашют. В какой момент скорость парашютиста станет постоянной, если радиус парашюта $R = 2,7$ м, масса парашютиста $m = 80$ кг.
4. Парашютист прыгает и сразу открывает парашют. В какой момент скорость парашютиста станет постоянной, если радиус парашюта $R = 2,5$ м, масса парашютиста $m = 70$ кг.
5. Парашютист прыгает и сразу открывает парашют. В какой момент скорость парашютиста станет постоянной, если радиус парашюта $R = 2$ м, масса парашютиста $m = 60$ кг.
6. Парашютист прыгает и сразу открывает парашют. В какой момент скорость парашютиста станет постоянной, если радиус парашюта $R = 2,2$ м, масса парашютиста $m = 65$ кг.
7. Парашютист прыгает и сразу открывает парашют. В какой момент скорость парашютиста станет постоянной, если радиус парашюта $R = 2,5$ м, масса парашютиста $m = 75$ кг.
8. В какой момент скорость спортсмена станет постоянной, если он прыгает с парашютом, имеющим форму шара, радиус которого $R = 3,5$ м, масса спортсмена $m = 80$ кг.
9. В какой момент скорость спортсмена станет постоянной, если он прыгает с парашютом, имеющим форму шара, радиус которого $R = 4,7$ м, масса спортсмена $m = 80$ кг.
10. В какой момент скорость спортсмена станет постоянной, если он прыгает с парашютом, имеющим форму шара, радиус которого $R = 4,2$ м, масса спортсмена $m = 70$ кг.

11. В какой момент скорость спортсмена станет постоянной, если он прыгает с парашютом, имеющим форму шара, радиус которого $R = 5,3 \text{ м}$, масса спортсмена $m = 65 \text{ кг}$.
12. В какой момент скорость спортсмена станет постоянной, если он прыгает с парашютом, имеющим форму шара, радиус которого $R = 6 \text{ м}$, масса спортсмена $m = 95 \text{ кг}$.
13. В какой момент скорость спортсмена станет постоянной, если он прыгает с парашютом, имеющим форму шара, радиус которого $R = 5,1 \text{ м}$, масса спортсмена $m = 80 \text{ кг}$.
14. В какой момент скорость спортсмена станет постоянной, если он прыгает с парашютом, имеющим форму шара, радиус которого $R = 5 \text{ м}$, масса спортсмена $m = 85 \text{ кг}$.
15. В какой момент скорость спортсмена станет постоянной, если он прыгает с парашютом, имеющим форму шара, радиус которого $R = 3,1 \text{ м}$, масса спортсмена $m = 80 \text{ кг}$.

10.2. Задача о движении тела, брошенного под углом к горизонту, с учетом сопротивления среды

Будучи брошенным под углом α к горизонту с начальной скоростью v_0 , тело летит без учета сопротивления воздуха по параболе и через некоторое время падает на землю.

При большой начальной скорости полета тела сопротивление воздуха может значительно изменить движение.

При учете силы сопротивления:

$$\vec{F}_{\text{сопр}} = k_1 \vec{v} + k_2 \vec{v}^2$$

получим уравнения движения в переменных v_x, v_y :

$$\begin{cases} \frac{dv_x}{dt} = -\frac{k_1 + k_2 \sqrt{(v_x^2 + v_y^2)}}{m} v_x \\ \frac{dv_y}{dt} = -g - \frac{k_1 + k_2 \sqrt{(v_x^2 + v_y^2)}}{m} v_y \end{cases} \quad (7)$$

Поскольку представляет интерес и траектория движения, дополним систему (7) еще двумя уравнениями:

$$\frac{dx}{dt} = v_x, \quad \frac{dy}{dt} = v_y, \quad (8)$$

Начальные

условия:

$$v_x(0) = v_0 \cos \alpha, \quad v_y(0) = v_0 \sin \alpha, \quad x(0) = 0, \quad y(0) = 0.$$

Решая системы (7) и (8) получим четыре функции:

$$v_x(t), \quad v_y(t), \quad x(t), \quad y(t). \quad \text{Системы (7) и (8) описывают}$$

движение с учетом сопротивления среды.

Проведем обезразмеривание задачи. Обезразмеривание заключается в переходе от абсолютных значений – расстояний, скоростей, времен (s , V , t) и т.д. – к относительным, причем отношения строятся к величинам типичным для данного движения.

Выполнив обезразмеривание во всех уравнениях систем (7), (8) получим систему дифференциальных уравнений для безразмерных переменных V_x , V_y , X , Y :

$$\begin{aligned} \frac{dV_x}{d\tau} &= -\underline{a} \cdot \sin \alpha \cdot V_x - \underline{b} \cdot \sin \alpha \cdot \sqrt{(V_x^2 + V_y^2)} \cdot V_x, \\ \frac{dX}{d\tau} &= \frac{V_x}{2 \cos \alpha}, \end{aligned} \quad (9)$$

$$\frac{dV_y}{d\tau} = -\sin \alpha - \underline{a} \cdot \sin \alpha \cdot V_y - \underline{b} \cdot \sin \alpha \cdot \sqrt{(V_x^2 + V_y^2)} \cdot V_y,$$

$$\frac{dY}{d\tau} = \frac{2V_y}{\sin \alpha},$$

где \underline{a} , \underline{b} -- безразмерные комбинации размерных параметров, входящих в исходные уравнения:

$$\underline{a} = \frac{k_1 v_0}{mg}, \quad \underline{b} = \frac{k_2 v_0^2}{mg}. \quad (10)$$

Начальные условия для безразмерных переменных имеют вид:

$$V_x(0) = \cos \alpha, \quad V_y(0) = \sin \alpha, \quad X(0) = 0, \quad Y(0) = 0 \quad (11)$$

Исследование задачи (9)-(11) при произвольных значениях угла α и коэффициентов \underline{a} , \underline{b} , выполним методом Рунге-Кутты с помощью программирования средствами VBA в Excel, чтобы составить ясное представление о влиянии линейной (коэффициент \underline{a}) и квадратичной (коэффициент \underline{b}) частей силы сопротивления на изучаемое движение.

Приведенная ниже программа VBA позволяет получить три графика:

1. первый представляет траекторию движения при учете линейной части силы сопротивления \underline{a} ($\underline{a} < > 0$, $\underline{b} = 0$),
2. второй представляет траекторию движения при учете квадратичной части силы сопротивления \underline{b} ($\underline{a} = 0$, $\underline{b} < > 0$),
3. третий представляет траекторию движения при учете обеих составляющих силы сопротивления \underline{a} и \underline{b} ($\underline{a} < > 0$, $\underline{b} < > 0$),

The dialog box 'UserForm1' contains the following elements:

- Step of integration: 0.01
- Launch angle: 45
- Instruction: Введите коэффициенты a и b в трех вариантах
- Input fields for coefficients:

a(1)	1	a(2)	0	a(3)	1
b(1)	0	b(2)	1	b(3)	1
- Buttons: Вычислить, Очистить, Заккрыть

Рис.10.3

Диалоговое окно для решения задачи методом Рунге-Кутта (рис.10.3) содержит поля ввода и управляющие кнопки:

1. шаг интегрирования метода Рунге-Кутта,
2. угол бросания α , $\alpha \in (0, 90^\circ)$,
3. входные данные \underline{a} , \underline{b} для решения задачи в трех вариантах:
 - Вариант1 – $\underline{a}(1) < > 0$, $\underline{b}(1) = 0$, (причем установленное в поле ввода нулевое значение не подлежит изменению, т. е. для текста ввода в свойство Enabled установлено False).
 - Вариант2 – $\underline{a}(2) = 0$, $\underline{b}(2) < > 0$,
 - Вариант3 – $\underline{a}(3) < > 0$, $\underline{b}(3) < > 0$,
 - Управляющие кнопки «Вычислить», «Очистить», «Заккрыть».

При нажатии на кнопку «Вычислить» на двух рабочих листах Excel будут получены результаты. На рабочем Листе 1 будет выведена таблица (рис.10.4) с результатами вычислений:

- В столбце А – получены значения времени t с заданным шагом интегрирования.
- В столбцах В, С – значения координат траектории $x(1)$, $y(1)$, полученных при первом варианте решения.

- В столбцах D, E – значения координат траектории $x(2)$, $y(2)$, полученных при втором варианте решения.
- В столбцах F, G – значения координат траектории $x(3)$, $y(3)$, полученных при третьем варианте решения.

	A	B	C	D	E	F	G	H	I	J	K
1											
2	t (время)	x(1)	y(1)	x(2)	y(2)	x(3)	y(3)				
3	0	0	0	0	0	0	0				
4	0,01	0,004982	0,01983	0,004982	0,01983	0,004965	0,01976				
5	0,02	0,00993	0,03932	0,00993	0,039323	0,009861	0,039047		Запуск программы		
6	0,03	0,014842	0,058475	0,014844	0,058482	0,014689	0,05787				
7	0,04	0,01972	0,077295	0,019724	0,077312	0,019452	0,076236				
8	0,05	0,024563	0,095783	0,024572	0,095816	0,02415	0,094156		Перейти на Лист2		
9	0,06	0,029373	0,113941	0,029387	0,113998	0,028785	0,111637				
10	0,07	0,034148	0,131773	0,034171	0,131861	0,033359	0,128687				
11	0,08	0,03889	0,149279	0,038924	0,149409	0,037872	0,145314				
12	0,09	0,043599	0,166463	0,043646	0,166645	0,042326	0,161525				
13	0,1	0,048274	0,183327	0,048339	0,183573	0,046723	0,177327				
14	0,11	0,052916	0,199872	0,053002	0,200196	0,051063	0,192728				
15	0,12	0,057526	0,216102	0,057636	0,216516	0,055348	0,207734				
16	0,13	0,062103	0,232018	0,062242	0,232537	0,059579	0,222351				
17	0,14	0,066648	0,247623	0,06682	0,248261	0,063757	0,236587				
18	0,15	0,071161	0,262918	0,07137	0,263693	0,067883	0,250447				
19	0,16	0,075642	0,277907	0,075894	0,278833	0,071958	0,263937				
20	0,17	0,080092	0,292591	0,080391	0,293686	0,075983	0,277064				
21	0,18	0,08451	0,306971	0,084863	0,308253	0,079959	0,289833				
22	0,19	0,088897	0,321052	0,089308	0,322538	0,083888	0,302248				
23	0,2	0,093253	0,334834	0,093729	0,336542	0,087769	0,314317				
24	0,21	0,097579	0,348319	0,098125	0,350269	0,091605	0,326044				
25	0,22	0,101874	0,36151	0,102496	0,36372	0,095395	0,337433				

Рис.10.4

На рабочем Листе 2 в Excel будут представлены три графика с траекториями, соответствующими трем вариантам решения (рис.10.5).

На рабочих листах Лист1 и Лист2 в Excel будут расположены по две командных кнопки.

На рабочем листе Лист1 расположены кнопки «Запуск программы» и «Перейти на Лист2». Создать процедуры обработки события нажатия этих кнопок:

Модуль рабочего листа Лист1

```
Private Sub CommandButton1_Click()
```

```
  Sheets("Лист1").Activate
```

```
  UserForm1.Show
```

```
End Sub
```

```
Private Sub CommandButton2_Click()
```

Sheets("Лист2").Activate
End Sub

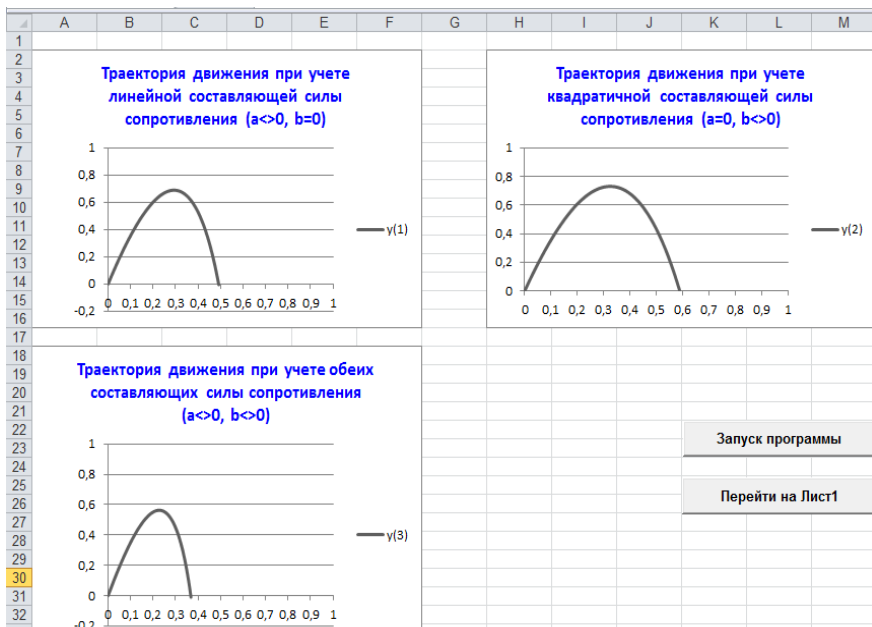


Рис.10.5

На рабочем листе Лист2 расположены кнопки «Запуск программы» и «Перейти на Лист1». Создать процедуры обработки события нажатия ЭТИХ кнопок:

Модуль рабочего листа Лист2

```
Private Sub CommandButton1_Click()
  Sheets("Лист 1").Select
  UserForm1.Show
End Sub
```

```
Private Sub CommandButton2_Click()
  Sheets("Лист 1").Select
End Sub
```

Код модуля формы

```
Const pi = 3.14
Dim s(1 To 3) As String
```

```

Dim a, b, C, vx, vy, dt, t As Single
Dim y0(0 To 3), y(0 To 3) As Single
Dim pa(1 To 3), pb(1 To 3) As Single
Dim p1, i, j, m, n, L, nom As Integer

```

' Ff - функция вычисления правых частей диф. уравнений системы (9)

```
Function Ff(i, y As Variant) As Single
```

```
Select Case i
```

```
Case 0 ' vx
```

```
Ff = -a * Sin(C) * y(0) - b * Sin(C) * (y(0) ^ 2 + y(1) ^ 2) ^ (1 / 2) * y(0)
```

```
Case 1 ' vy
```

```
Ff = -Sin(C) - a * Sin(C) * y(1) - b * Sin(C) * (y(0) ^ 2 + y(1) ^ 2) ^ (1 / 2) * y(1)
```

```
Case 2 ' x
```

```
Ff = y(0) / (2 * Cos(C))
```

```
Case 3 ' y
```

```
Ff = 2 * y(1) / Sin(C)
```

```
End Select
```

```
End Function ' конец Ff
```

' right - вычисление набора значений правой части

```
Sub right(y, F As Variant)
```

```
For j = 0 To n - 1
```

```
F(j) = Ff(j, y)
```

```
Next
```

```
End Sub ' конец right
```

' rk4 - метод Рунге-Кутты четвертого порядка

```
Sub rk4(dt, y0, y As Variant)
```

```
Dim z(0 To 3), k1(0 To 3), k2(0 To 3), k3(0 To 3), k4(0 To 3) As Single
```

```
Call right(y0, k1)
```

```
For i = 0 To n - 1
```

```
z(i) = y0(i) + k1(i) * dt / 2
```

```
Next
```

```
Call right(z, k2)
```

```
For i = 0 To n - 1
```

```
z(i) = y0(i) + k2(i) * dt / 2
```

```
Next
```

```

Call right(z, k3)
For i = 0 To n - 1
z(i) = y0(i) + k3(i) * dt
Next
Call right(z, k4)
For i = 0 To n - 1
y(i) = y0(i) + (1 / 6) * dt * (k1(i) + 2 * k2(i) + 2 * k3(i) + k4(i))
Next
End Sub 'конец rk4

```

' Процедура инициализации формы с помощью метода Show

```

Private Sub userform_initialize()
TextBox4.Text = 0
TextBox5.Text = 0
TextBox4.Enabled = False
TextBox5.Enabled = False
Sheets("Лист 1").Activate
End Sub

```

' Процедура обработки события нажатия кнопки «Вычислить»

```

Private Sub CommandButton1_Click()
C = Val(TextBox1.Text) * pi / 180
dt = Val(TextBox2.Text)
pa(1) = Val(TextBox3.Text)
pb(1) = Val(TextBox4.Text)
pa(2) = Val(TextBox5.Text)
pb(2) = Val(TextBox6.Text)
pa(3) = Val(TextBox7.Text)
pb(3) = Val(TextBox8.Text)
n = 4: m = 1: x = 0
For L = 1 To 3          ' начало цикла по номерам вариантов
a = pa(L): b = pb(L)
y0(0) = Cos(C): y0(1) = Sin(C): y0(2) = 0: y0(3) = 0: t = 0
Cells(3, L + m).Activate
ActiveCell.Value = y0(2)
ActiveCell.Offset(0, 1).Value = y0(3)
If L = 1 Then ActiveCell.Offset(0, -1).Value = t
ActiveCell.Offset(1, 0).Activate

```



```

y(3) = y0(3)
Do While y(3) >= 0 ' начало цикла расчетов пока y>= 0
t = t + dt
Call rk4(dt, y0, y):
For i = 0 To n - 1: y0(i) = y(i): Next
ActiveCell.Value = y(2)
ActiveCell.Offset(0, 1).Value = y(3)
If L = 1 Then
ActiveCell.Offset(0, -1).Value = t
End If
ActiveCell.Offset(1, 0).Activate
Loop ' конец цикла пока
' Запоминаем адрес правого нижнего угла каждого диапазона,
' по которым будут строиться диаграммы
s(L) = ActiveCell.Offset(-1, 1).Address(rowabsolute:=False,
columnabsolute:=False)
m = m + 1
Next L ' конец цикла по вариантам
' Выделяем диапазон с результатами вычислений x(1), y(1)
Set r = Sheets("Лист 1").Range("B2:" + s(1))
r.Select
Call Grafic1
Sheets("Лист 1").Select
' Выделяем диапазон с результатами вычислений x(2), y(2)
Set r = Sheets("Лист 1").Range("D2:" + s(2))
r.Select
Call Grafic2
Sheets("Лист 1").Select
' Выделяем диапазон с результатами вычислений x(3), y(3)
Set r = Sheets("Лист 1").Range("F2:" + s(3))
r.Select
Call Grafic3
End Sub ' конец CommandButton1_Click()

' Процедура обработки события нажатия кнопки «Очистить»
Private Sub CommandButton3_Click()
Sheets("Лист 1").Select
nom = Cells(3, 1).CurrentRegion.Rows.Count

```

```

Range(Cells(3, 1), Cells(nom + 1, 7)).Clear
'
With Worksheets("Лист 2")
If .ChartObjects.Count > 0 Then
    .ChartObjects.Delete
End If
End With
End Sub 'конец CommandButton3_Click()

```

```

'Процедура обработки события нажатия кнопки «Заккрыть»
Private Sub CommandButton2_Click()
UserForm1.Hide
End Sub 'конец CommandButton2_Click()

```

Для создания макросов построения диаграмм необходимо выполнить следующие действия:

1. Выделить соответствующий диапазон. Перейти на вкладку **Разработчик** и в группе **Код** нажать на кнопку **Запись макроса**. В открывшемся окне **Запись макроса** ввести название макроса.

2. Перейти на вкладку **Вставка**. В группе **Диаграммы** выбрать тип графика «Точечная», вид графика и стиль линий.

3. Перейти на вкладку **Макет**, в группе **Оси** установить параметры основной горизонтальной оси: *максимальное значение 1, цена основных делений 0.1*; вертикальной оси: *максимальное значение 1, цена основных делений 0.2* (рис. 10.6 и 10.7).

4. На вкладке **Макет**, в группе **Подписи** нажать кнопку **Заголовок диаграммы**, в открывшемся окне выбрать **Над диаграммой**, в области диаграммы ввести название диаграммы, перейти на вкладку **Главная**, в группе **Шрифт** уменьшить размер шрифта.

5. На вкладке **Разработчик** в группе **Код** нажать кнопку **Остановить запись**.

6. Проверить работу макроса. Удалить диаграмму с рабочего листа, перейти на вкладку **Разработчик** в группе **Код** нажать кнопку **Макросы**, в открывшемся окне выбрать созданный макрос и нажать кнопку **Выполнить**.

7. На вкладке **Разработчик** в группе **Код** нажать кнопку **Макросы**, в открывшемся окне выбрать созданный макрос и нажать кнопку **Изменить**. Внести изменения в программу:

а) Каждую диаграмму разместить на определенном месте рабочего листа **Лист2**. Для этого в макросы создания диаграмм вставить команды, которые размещают диаграмму на рабочем листе **Лист2**, определяют положение верхней кромки, левой кромки, ширины и высоты диаграммы:

Размещение первого графического объекта на рабочем листе **Лист2** в диапазоне A2:F16

```
ActiveChart.Location Where:=xlLocationAsObject, Name:="Лист2"  
With ActiveSheet.ChartObjects(1)  
.Top = Range("A2").Top  
.Left = Range("A2").Left  
.Width = Range("A2:F16").Width  
.Height = Range("A2:F16").Height  
End With
```

Размещение второго графического объекта на рабочем листе **Лист2** в диапазоне H2:M16

```
ActiveChart.Location Where:=xlLocationAsObject, Name:="Лист2"  
With ActiveSheet.ChartObjects(2)  
.Top = Range("H2").Top  
.Left = Range("H2").Left  
.Width = Range("H2:M16").Width  
.Height = Range("H2:M16").Height  
End With
```

Размещение третьего графического объекта на рабочем листе **Лист2** в диапазоне A18:F33

```
ActiveChart.Location Where:=xlLocationAsObject, Name:="Лист2"  
With ActiveSheet.ChartObjects(3)  
.Top = Range("A18").Top  
.Left = Range("A18").Left  
.Width = Range("A18:F33").Width  
.Height = Range("A18:F33").Height  
End With
```

Параметры оси

минимальное значение: авто фиксированное 0,0

максимальное значение: авто фиксированное 1

цена основных делений: авто фиксированное 0,1

цена промежуточных делений: авто фиксированное 0,02

обратный порядок значений

логарифмическая шкала Основная: 10

Цена деления: нет ▾

Отображать на диаграмме

Рис. 10.6 Параметры горизонтальной оси

Параметры оси

минимальное значение: авто фиксированное -0,1

максимальное значение: авто фиксированное 1

цена основных делений: авто фиксированное 0,2

цена промежуточных делений: авто фиксированное 0,02

обратный порядок значений

логарифмическая шкала Основная: 10

Цена деления: нет ▾

Отображать на диаграмме

Рис. 10.7 Параметры вертикальной оси

б) При изменении исходных данных, диапазон данных, по которому строится диаграмма, тоже меняется. Для того чтобы перестраивать диаграмму при изменении диапазона, в общей части (General) модуля макросов описать глобальную переменную `r` типа Range:

```
Public r As Range
```

Диапазон, по которому будет строиться диаграмма, устанавливается в модуле формы в процедуре `CommandButton1_Click`:

```
Set r = Sheets("Лист 1").Range("B2:" + s(1))
```

`r.Select` ' для первого графического объекта

Для остальных графических объектов диапазоны устанавливаются аналогично.

В макросах для `SetSourceData` установить диапазон `r`.

```
ActiveChart.SetSourceData Source:=r
```

в) При необходимости в области описания заголовка можно изменить сам заголовок, но при этом нужно изменить и число символов заголовка – в свойстве `Characters` второй параметр

(например, Characters(1, 82)). Можно изменить цвет и размер шрифта (ForeColor.RGB и Size).

Макрос Grafic1 построения первого графического объекта имеет вид:

```
Public r As Range
```

```
Sub Grafic1()
```

```
' Grafic1 Макрос '
```

```
ActiveSheet.Shapes.AddChart.Select
```

```
ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
```

```
ActiveChart.SetSourceData Source:=r 'установка диапазона
```

```
ActiveChart.ChartStyle = 1 'стиль линий
```

```
ActiveChart.ChartTitle.Select
```

```
ActiveChart.ChartTitle.Text = "Траектория движения при учете  
линейной составляющей силы сопротивления (a<>0, b=0)"
```

```
Selection.Format.TextFrame2.TextRange.Characters.Text = "  
Траектория движения при учете линейной составляющей силы  
сопротивления (a<>0, b=0)"
```

```
With Selection.Format.TextFrame2.TextRange.Characters(1,  
82).ParagraphFormat
```

```
.TextDirection = msoTextDirectionLeftToRight
```

```
.Alignment = msoAlignCenter
```

```
End With
```

```
With Selection.Format.TextFrame2.TextRange.Characters(1, 82).Font
```

```
.BaselineOffset = 0
```

```
.Bold = msoTrue
```

```
.NameComplexScript = "+mn-cs"
```

```
.NameFarEast = "+mn-ea"
```

```
.Fill.Visible = msoTrue
```

```
.Fill.ForeColor.RGB = RGB(0, 0, 255) 'цвет
```

```
.Fill.Transparency = 0
```

```
.Fill.Solid
```

```
.Size = 13 'размер шрифта
```

```
.Italic = msoFalse
```

```
.Kerning = 12
```

```
.Name = "+mn-lt"
```

```
.UnderlineStyle = msoNoUnderline
```

```
.Strike = msoNoStrike
```

```

End With
    ' установка параметров осей
    ActiveChart.Axes(xlCategory).Select
    ActiveChart.Axes(xlCategory).MaximumScale = 1
    ActiveChart.Axes(xlCategory).MajorUnit = 0.1
ActiveChart.HasAxis(xlValue) = True
    ActiveChart.Axes(xlValue).Select
    ActiveChart.Axes(xlValue).MaximumScale = 1
    ActiveChart.Axes(xlValue).MajorUnit = 0.2
    ActiveChart.PlotArea.Select
' размещение графического объекта на рабочем листе Лист2 в
диапазоне A2:F16
ActiveChart.Location Where:=xlLocationAsObject, Name:="Лист 2"
With ActiveSheet.ChartObjects(1)
.Top = Range("A2").Top
.Left = Range("A2").Left
.Width = Range("A2:F16").Width
.Height = Range("A2:F16").Height
End With
End Sub

```

Задания для самостоятельного выполнения

Задание №1

Рассмотрим полет чугунного ядра радиуса $R=0,1$ м, выпущенного с начальной скоростью $v_0 = 60$ м/с под углом $\alpha = 45^\circ$ к поверхности Земли. Определим, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется. Будем решать обезразмеренные уравнения (9), чтобы сократить число параметров. Вычислим значения параметров \underline{a} и \underline{b} с помощью формул (10), после чего решим систему дифференциальных уравнений (9) с начальными условиями (11) методом Рунге-Кутты. Учтем, что плотность чугуна $\rho_{\text{чуг}} = 7800$ кг/м³.

$$\underline{a} = \frac{k_1 v_0}{mg} = \frac{6\pi\mu R v_0}{\frac{4}{3}\pi R^3 \rho_{\text{чуг}} g} = \frac{9\mu v_0}{2r^2 \rho_{\text{чуг}} g} \approx 0,0063,$$

$$\underline{b} = \frac{k_2 v_0^2}{mg} = \frac{\frac{1}{2} C \pi R^2 \rho_{\text{возд}} v_0^2}{\frac{4}{3} \pi R^3 \rho_{\text{чуг}} g} = \frac{3C \rho_{\text{возд}} v_0^2}{8R \rho_{\text{чуг}} g} \approx 0,034.$$

Расчеты провести с шагом интегрирования 0.1, и получить траектории движения для трех вариантов исследования:

1. при учете только линейной части силы сопротивления a,
2. при учете только квадратичной части силы сопротивления b,
3. при учете обеих составляющих силы сопротивления a и b.

Расчеты надо проводить до тех пор, пока ядро не достигнет земли, т.е. пока значение Y не станет равным 0 (в программе это цикл Do While $y(3) >= 0$).

Для вариантов самостоятельного выполнения дается таблица:

Таблица 10.2 Таблица плотности разных веществ

<i>Вещество</i>	$\rho, \text{кг/м}^3$	<i>Вещество</i>	$\rho, \text{кг/м}^3$
Воздух	1,29	Олово	$7,3 \cdot 10^3$
Резина	$1,2 \cdot 10^3$	Титан	$4,5 \cdot 10^3$
Алюминий	$2,7 \cdot 10^3$	Сталь	$7,7 \cdot 10^3$
Серебро	$10,5 \cdot 10^3$	Никель	$8,8 \cdot 10^3$
Чугун	$7,8 \cdot 10^3$	Золото	$19,3 \cdot 10^3$
Кварцевое стекло	$2,21 \cdot 10^3$	Медь	$8,89 \cdot 10^3$

Задание №2

1. Рассмотреть полет чугунного ядра радиуса $R=0,3$ м, выпущенного с начальной скоростью $v_0 = 50$ м/с под углом $\alpha = 10^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
2. Рассмотреть полет серебряного ядра радиуса $R=0,5$ м, выпущенного с начальной скоростью $v_0 = 40$ м/с под углом $\alpha = 35^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
3. Рассмотреть полет кварцевого ядра радиуса $R=0,3$ м, выпущенного с начальной скоростью $v_0 = 30$ м/с под углом $\alpha = 15^\circ$

- к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
4. Рассмотреть полет титанового ядра радиуса $R=0,1$ м, выпущенного с начальной скоростью $v_0 = 40$ м/с под углом $\alpha = 65^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
 5. Рассмотреть полет стального ядра радиуса $R=0,3$ м, выпущенного с начальной скоростью $v_0 = 10$ м/с под углом $\alpha = 30^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
 6. Рассмотреть полет медного ядра радиуса $R=0,8$ м, выпущенного с начальной скоростью $v_0 = 70$ м/с под углом $\alpha = 75^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
 7. Рассмотреть полет резинового ядра радиуса $R=0,7$ м, выпущенного с начальной скоростью $v_0 = 60$ м/с под углом $\alpha = 25^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
 8. Рассмотреть полет никелевого ядра радиуса $R=0,1$ м, выпущенного с начальной скоростью $v_0 = 80$ м/с под углом $\alpha = 65^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
 9. Рассмотреть полет чугунного ядра радиуса $R=0,2$ м, выпущенного с начальной скоростью $v_0 = 40$ м/с под углом $\alpha = 25^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
 10. Рассмотреть полет золотого ядра радиуса $R=0,1$ м, выпущенного с начальной скоростью $v_0 = 100$ м/с под углом $\alpha = 30^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
 11. Рассмотреть полет оловянного ядра радиуса $R=0,5$ м, выпущенного с начальной скоростью $v_0 = 70$ м/с под углом $\alpha = 30^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
 12. Рассмотреть полет стального ядра радиуса $R=0,2$ м, выпущенного с начальной скоростью $v_0 = 80$ м/с под углом $\alpha = 75^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.

13. Рассмотреть полет серебряного ядра радиуса $R=0,4$ м, выпущенного с начальной скоростью $v_0 = 100$ м/с под углом $\alpha = 45^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
14. Рассмотреть полет стального ядра радиуса $R=0,1$ м, выпущенного с начальной скоростью $v_0 = 100$ м/с под углом $\alpha = 10^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.
15. Рассмотреть полет алюминиевого ядра радиуса $R=0,6$ м, выпущенного с начальной скоростью $v_0 = 80$ м/с под углом $\alpha = 60^\circ$ к поверхности Земли. Определить, какое расстояние пролетит ядро, на какую максимальную высоту оно поднимется.

Литература

1. *А.Ю. Гарнаев, Л.В. Рудикова* Microsoft Office Excel 2010: разработка приложений. – СПб.: БХВ-Петербург, 2011. – 528с.
2. *А. Гарнаев*. Самоучитель VBA. – Санкт-Петербург, Изд-во «БХВ-Петербург», 2002. – 512с.
3. *Ларсен Рональд*. Инженерные расчеты в Excel. – М.: Изд-во «Вильямс», 2004. – 544с.
4. *Фризен И.Г.* Офисное программирование. Учебное пособие. – Ростов-на-Дону: 2010. – 240 с.
5. *Могилев А.В., Пак Н.И., Хеннер Е.К.* Практикум по информатике: Учеб. пособие для студ. пед. вузов/ Под ред. Е.К. Хеннера. – М.: Изд. центр "Академия", 2001. -608с.
6. *Широкова О. А., Хрусталева А, В.* Практикум по компьютерному моделированию. Часть II: Компьютерное моделирование физических процессов. – Казань, ИЦ КГУ, 2009 – 48с.
7. *Насырова Н.Х., Бахтиева Л.У.* Microsoft Word, Excel, Access, язык HTML для студентов гуманитарных факультетов: учебное пособие – Казань: Изд-во Казанского университета, 2011. – 84с.