

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«РЕШЕНИЕ ЗАДАЧ В MS EXCEL
С ИСПОЛЬЗОВАНИЕМ VBA»
Методические указания и примеры**

(для студентов всех специальностей)

Рассмотрено
на заседании кафедры ВМиП
протокол №7 от 12 февраля 2007 г.

Утверждено
методической комиссией ДонНТУ
протокол №5 от 28 февраля 2007 г.
Регистрационный номер №6131

2007

УДК 681.3.06(071)

Р 31

«Решение задач в MS EXCEL с использованием VBA»: Методические указания и примеры / Авторы: Зензеров В.И., Славинская Л.В., Ефименко К.Н., Добровольский Ю.Н. – Донецк: ДонНТУ, 2007. – 55 с.

Методические указания посвящены вопросам разработки программных модулей в среде Visual Basic for Application для обработки информации и решения экономических задач в MS Excel. Содержат материал по основам программирования в среде VBA и примеры проектов для выполнения лабораторных работ, что способствует практическому усвоению материала.

Материал полезен для студентов всех специальностей, изучающих курс «Информатика и КТ», а также для преподавателей, аспирантов и других пользователей персональных компьютеров.

Авторы:

Зензеров В.И., ст. преподаватель,

Славинская Л.В., ассистент,

Ефименко К.Н., ассистент,

Добровольский Ю.Н., ст. преподаватель

Отв. за выпуск:

д.т.н., проф. Павлыш В.Н.

СОДЕРЖАНИЕ

Введение.....	4
1. Интегрированная среда разработки приложений на языке VBA.....	5
2. Особенности программирования на языке VBA.....	8
3. Создание экранных форм и включение их в проекты.....	10
4. Структура программы. Модули, процедуры и функции.....	19
5. Основные операторы языка VBA.....	23
6. Примеры разработки проектов в среде VBA	35
Заключение.....	55
Список литературы.....	56

ВВЕДЕНИЕ

Мечта пользователей персонального компьютера иметь универсальное, мощное, гибкое и удобное средство, позволяющее быстро и эффективно решать самые разнообразные задачи, в том числе экономические и финансовые, воплотилась в MS ECEL. Visual Basic for Application (VBA) существенно обогатил и усилил MS ECEL. VBA – это сочетание одного из самых простых языков программирования и всех вычислительных возможностей MS ECEL. С помощью VBA вы сможете легко и быстро создавать разнообразные приложения, даже не являясь специалистом в области программирования. VBA содержит графическую среду, позволяющую наглядно конструировать экранные формы из управляющих элементов. VBA в сочетании с возможностями MS ECEL позволяет решать задачи, о решении которых только средствами MS ECEL ранее даже и не говорили.

Настоящее методическое пособие поможет вам грамотно и эффективно использовать компьютер в своей работе, изучить особенности программирования на языке Visual Basic for Application для обработки данных, представленных в виде таблиц в MS ECEL, и, таким образом, расширить возможности стандартных приложений (в частности, популярного пакета MS Office). Расширение возможностей стандартных компонентов пакета Microsoft Office для решения конкретных задач достигается созданием собственных приложений путем программирования на языке VBA, который доступен любому квалифицированному пользователю персональных компьютеров.

Изложенный в пособии методический материал с примерами предполагают, что пользователь знаком с основами работы в приложении MS Excel и желает расширить свои знания в области применения VBA для решения разнообразных задач.

1. ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ ПРИЛОЖЕНИЙ НА ЯЗЫКЕ VBA

Для вызова интегрированной среды разработки приложений (IDE) необходимо, выбрать в меню следующие команды: **Сервис** → **Макрос** → **Редактор Visual Basic**. Общий вид IDE приведен на рис. 1.1. IDE состоит из нескольких компонентов: главного меню, панели инструментов, окна проекта, окна свойств, панели элементов, конструктора форм, окна контрольных значений и нескольких других вспомогательных окон.

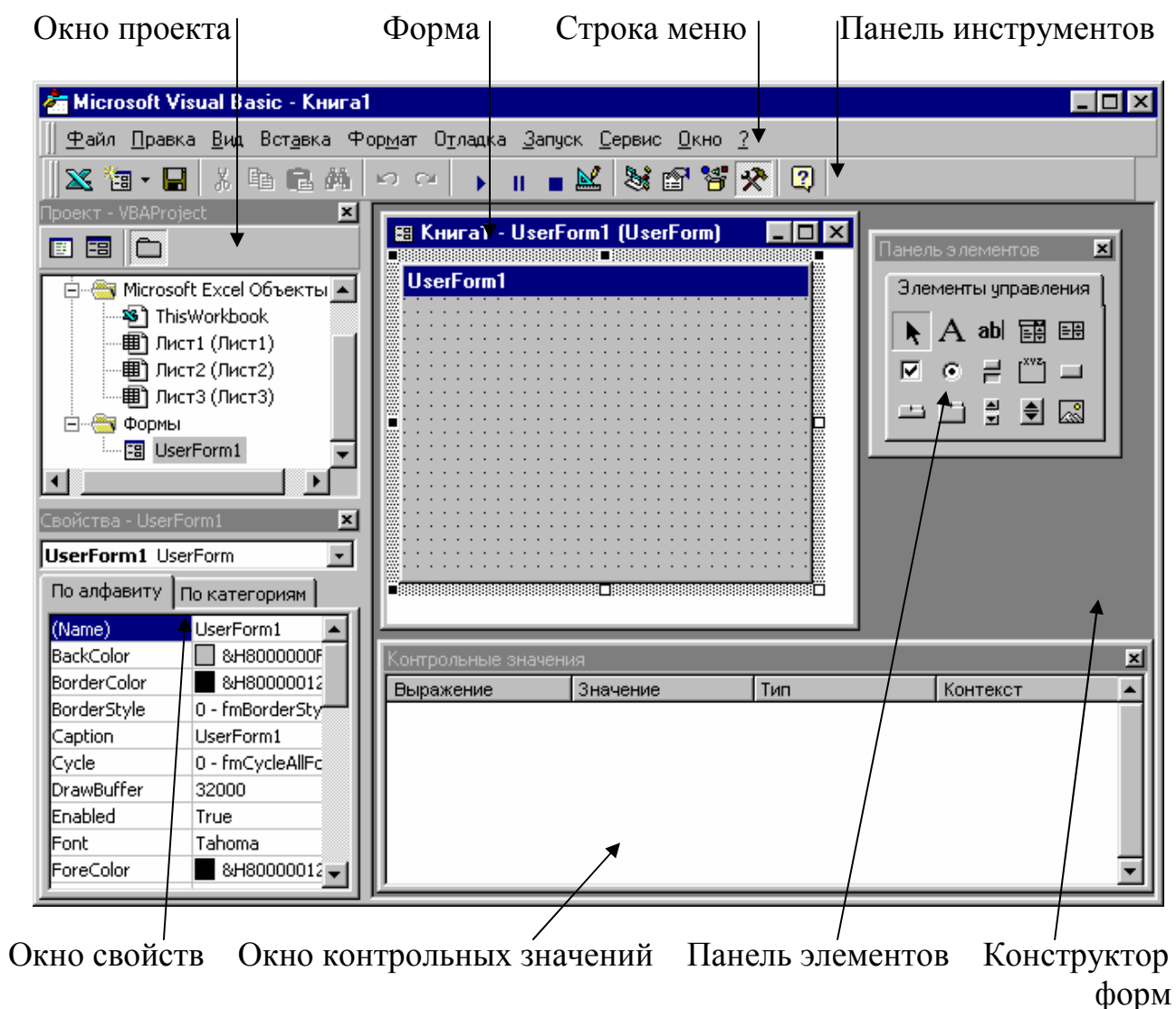


Рис. 1.1. Общий вид IDE VBA

Главное меню – это строка текста, расположенная в верхней части окна Visual Basic, и состоит из следующих пунктов:

✓ **Файл** – предназначено для работы с файлами, из которых образуются приложения. В нем можно создавать, сохранять и печатать проекты.

- ✓ **Правка** – выполняет стандартные операции с буфером обмена – вырезание, копирование и вставка. Они применяются не только к фрагментам программы, но и к управляющим элементам.
- ✓ **Вид** – включаются режимы просмотра различных компонентов и инструментов. Просматривать можно формы и программные модули.
- ✓ **Вставка** – позволяет добавлять процедуры, формы, модули и модули класса.
- ✓ **Формат** – содержит команды, определяющие расположение и размеры элементов и форм.
- ✓ **Отладка** – содержит команды, позволяющие запустить и остановить приложение, расставить точки прерывания и выбрать просматриваемые объекты, а также выполнить другие операции, помогающие следить за работой приложения.
- ✓ **Запуск** – содержит команды, которые запускают и останавливают приложение, прерывают и возобновляют выполнение программы, что особенно удобно в процессе отладки.
- ✓ **Сервис** – позволяет включить дополнительные элементы, запустить макросы и настроить параметры редактора.
- ✓ **Окно** – позволяет выстроить окна IDE (каскадное или мозаичное расположение), упорядочить значки свернутых форм, а также создает список, позволяющий быстро перейти к одному из открытых окон IDE.
- ✓ **?** – помощь пользователю. Для быстрого вызова главного меню необходимо нажать клавишу F10.

Панель инструментов находится под главным меню (рис. 1.2). Если она отсутствует, необходимо выполнить команду **Вид** → **Панели инструментов** → **Стандарт**.

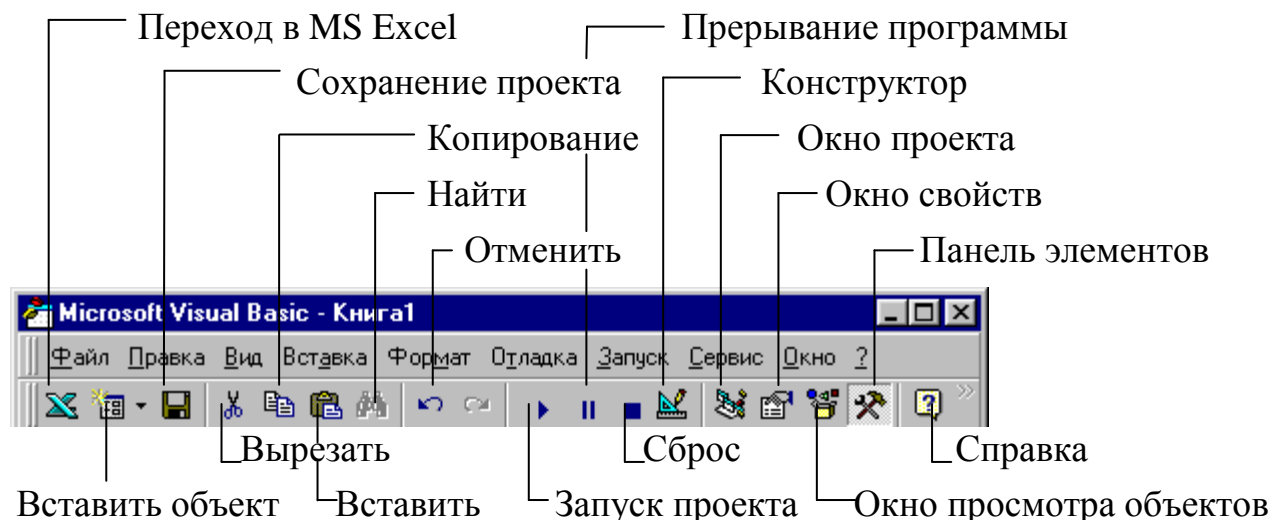


Рис. 1.2. Панель инструментов VBA

Окно проекта напоминает собой окно Проводника Windows, и предназначено для быстрого просмотра составляющих проекта, который объединяет в себе все объекты, составляющие приложение (рис. 1.3). Это стандартные объекты открытого приложения Microsoft Office (документ редактора MS Word, книга и её листы MS Excel), формы, модули и классы.

Окно свойств отображает различные атрибуты выделенного объекта (рис. 1.4). Все объекты (формы, управляющие элементы и т.д.) имеют атрибуты, которые изменяют не только внешний вид объекта, но и его поведение. Все эти атрибуты называются свойствами. Следовательно, каждый объект обладает набором свойств.

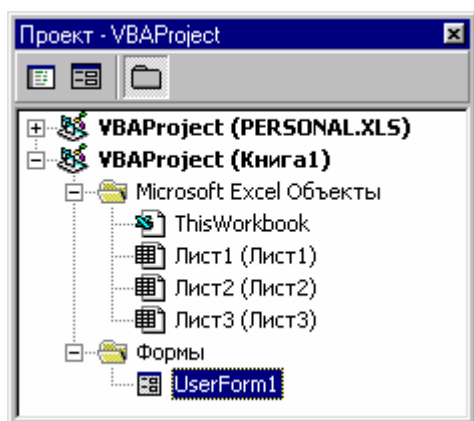


Рис. 1.3. Окно проекта

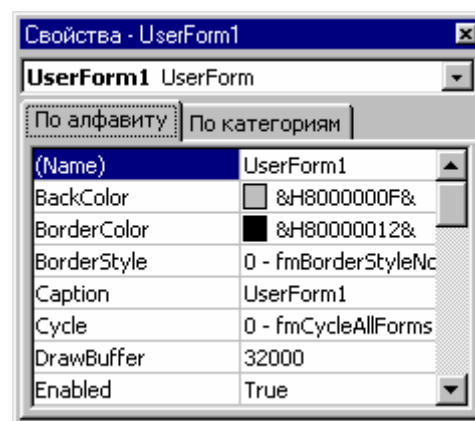



Рис. 1.4. Окно свойств

Окно контрольных значений позволяет просматривать значения контрольных переменных в процессе проверки правильности работы (отладки) проекта, что позволяет находить ошибки в логике работы программ.

Конструктор форм расположен в центре экрана редактора VBA. Здесь выводится либо изображение формы, что позволяет производить визуальное конструирование макета формы и расположенных на ней элементов, либо окно программы (более подробно об этом будет рассказано ниже).

2. ОСОБЕННОСТИ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ VBA

Процесс разработки программы на языке VBA – **проекта**, может состоять из нескольких этапов, в зависимости от конечного результата. Если необходимо получить программу, которая будет производить определенные вычисления или действия, расширяющие математические возможности стандартного приложения Microsoft Office, то достаточно создать **программный модуль**. Для применения этой программы можно поместить в рабочей области приложения кнопку, нажатие которой будет вызывать выполнение программы. Для этого в приложении необходимо включить панель инструментов с помощью команды **Вид → Панели инструментов → Элементы управления**, а затем создать кнопку с соответствующим программным кодом. Либо выполнять программу с помощью команды **Сервис → Макрос → Макросы**.

Разработка “полноценной” программы (для выполнения которой требуется отдельное окно, с различными элементами управления) будет включать два этапа. Первый этап – этап визуального программирования, на котором создается окно (**форма**) программы, где располагаются необходимые элементы управления. Второй – этап программирования, на котором создаются части программы (**процедуры**), выполняющиеся в ответ на определенные события. Событием является, например, щелчок левой кнопкой мыши на командной кнопке (событие Click), нажатие клавиши на клавиатуре (событие KeyPress) и т.д. Использовать такое приложение можно нажатием кнопки  – «Запуск проекта».

2.1. Объекты, свойства и методы VBA

Одним из основных понятий VBA является объект. **Объект** – это то, чем вы управляете с помощью программы на языке VBA, например, форма, кнопка, рабочий лист или диапазон ячеек MS Excel. Каждый объект обладает некоторыми **свойствами**. Например, форма может быть видимой или невидимой в данный момент на экране. Другой пример свойства объекта – шрифт для отображения информации в ячейке (объекте) рабочего листа.

Объект содержит также список методов, которые к нему применимы. **Методы** – это то, что вы можете делать с объектом. Например, показать форму на экране или убрать её можно с помощью методов Show и Hide.

Таким образом, **объект** – это программный элемент, который имеет свое отображение на экране, содержит некоторые переменные, определяющие его **свойства**, и некоторые **методы** для управления объектом. Например, в MS Excel имеется много встроенных объектов:

Range (“Адрес”)	- диапазон ячеек (может включать только одну ячейку).
Cells (i, j)	- ячейка, находящаяся на пересечении i-й строки и j-го столбца рабочего листа MS Excel (i и j – целые числа).
Rows (№ строки)	- строка с заданным номером.
Columns (№ столбца)	- столбец с заданным номером
Sheets (“Имя”)	- лист с указанным именем.
Sheets (№ листа)	- лист с указанным номером.
WorkSheet	- рабочий лист.

Установка значений свойств – это один из способов управления объектами. Синтаксис установки значения свойства объекта следующий:

Объект. Свойство = Выражение

Основным свойством объектов **Cells** и **Range**, является **Value** (значение), которое, однако, можно не указывать. Например:

Range(“A5:A10”).**Value** = 0 или **Range**(“A5:A10”) = 0 – в диапазон ячеек A5:A10 заносится значение 0.

Cells(2, 4).**Value** = n или **Cells**(2, 4) = n – в ячейку, находящуюся на пересечении 2-й строки и 4-го столбца (ячейка с адресом “D2”), заносится значение переменной n.

Синтаксис чтения свойств объекта следующий:

Переменная = Объект. Свойство

Например:

Xn = Cells(1, 2).**Value** или **Xn = Range**(“B1”).**Value** – переменной Xn присваивается значение из ячейки B1 текущего рабочего листа.

Синтаксис применения методов к объекту:

Объект. Метод

Например:

Sheets(2).**Activate** – сделать активным лист с №2.

Sheets(“Диаграмма”).**Delete** – удалить лист с именем “Диаграмма”.

Range(“A5:A10”).**Clear** – очистить диапазон ячеек A5:A10.

Range(“A2:B10”).**Select** – выделить диапазон ячеек A2:B10.

В MS Excel имеются объекты, которые содержат другие объекты. Например, рабочая книга содержит рабочие листы, рабочий лист содержит диапазон ячеек и т.д. Объектом самого высокого уровня является **Application** (приложе-

ние). Если вы изменяете его свойства или вызываете его методы, то результат применяется к текущей работе MS Excel. Например:

Application.Quit - завершение работы с Excel.

Отметим, что точка после имени объекта может использоваться для перехода от одного объекта к другому. Например, следующее выражение очищает вторую строку рабочего листа **Май** в рабочей книге **Отчет**:

Application.Workbooks("Отчет").Worksheets("Май").Rows(2).Delete

Нужно отметить следующее:

- Можно не писать имя объекта **Application**, так как это подразумевается по умолчанию.
- При работе с подобъектом уже активизированного объекта нет необходимости указывать содержащий его объект.
- VBA использует некоторые свойства и методы, которые возвращают объект к которому они относятся (это позволяет быстро указывать нужный объект). Примеры таких свойств: **ActiveCell** (активная ячейка), **ActiveSheet** (активный лист), **ActiveWorkBook** (активная рабочая книга). Так, установить значение активной ячейки можно следующим образом:

ActiveCell.Value = " Да".

3. СОЗДАНИЕ ЭКРАННЫХ ФОРМ И ВКЛЮЧЕНИЕ ИХ В ПРОЕКТЫ

3.1. Создание форм. Свойства, события и методы форм

Форма – это главный объект, образующий визуальную основу приложения. По своей сути форма представляет собой окно, в котором можно размещать различные управляющие элементы при создании приложений. Для создания формы необходимо выполнить команду **Вставка → UserForm**. В окне Конструктора

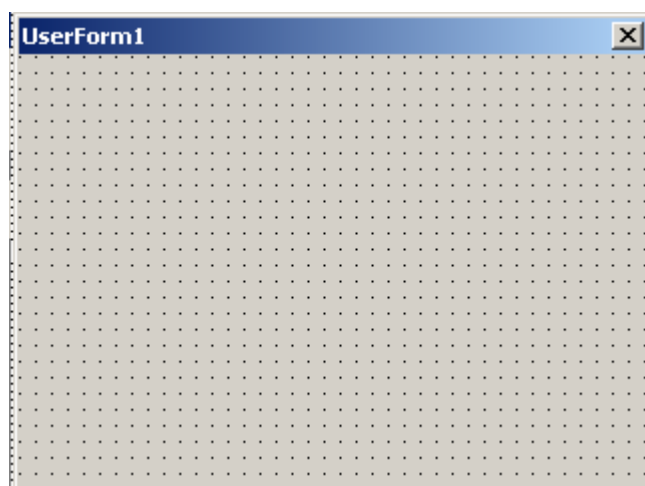


Рис. 3.1. Объект форма

форм появится форма, имеющая стандартный вид для ОС Windows (рис. 3.1).

Как и любой другой объект VBA форма имеет набор **свойств**, основные из которых приведены в таблице 3.1. Для получения справки по любому свойству достаточно выделить его в окне свойств и нажать F1.

Таблица 3.1. Основные свойства формы

Свойство	Описание
<i>BackColor</i>	Цвет фона для формы.
<i>BorderStyle</i>	Определяет тип границы, окружающей форму
<i>Caption</i>	Текст, который выводится в заголовке формы.
<i>Font</i>	Определяет тип и вид шрифта в форме.
<i>Height</i>	Определяет высоту формы в твипах.
<i>(Name)</i>	Имя объекта, для программы VBA.
<i>Width</i>	Определяет ширину формы в твипах.

Свойства можно изменять в режиме конструирования в окне свойств, либо программно в режиме выполнения. Например, в ходе выполнения программы можно изменить заголовок формы командой:

frmForm1.Caption = "Привет"

Программы в ОС Windows управляются **событиями**. Каждый раз, когда нажимается кнопка, перемещается мышь, изменяются размеры формы и т.д., ОС генерирует сообщение. Сообщение доставляется соответствующему объекту, например форме, а та генерирует соответствующее событие. Следовательно, можно составить фрагмент программы, в котором объект будет реагировать на событие определенным образом, т.е. любому стандартному событию соответствует определенная процедура. Чтобы просмотреть события связанные с формой, необходимо в режиме конструирования дважды щелкнуть на ней – появится окно программы, в котором щелкнуть на списке Процедура. В табл. 3.2 приведены наиболее часто используемые события.

Таблица 3.2. Основные события форм

Событие	Описание
<i>Initialize</i>	Происходит во время конфигурации и до загрузки формы в память.
<i>Activate</i>	Происходит после загрузки формы в память.
<i>Deactivate</i>	Происходит, если форма перестает быть активной.
<i>Click</i>	Происходит при нажатии левой кнопки мыши на форме.

Следующий пример изменяет заголовок формы при активизации, и уменьшает размер формы после щелчка левой кнопкой мыши на форме.

```

Private Sub UserForm_Activate()
    frmForm1.Caption = "Щелчок на форме уменьшает её размеры"
End Sub
Private Sub UserForm_Click()
    frmForm1.Width = frmForm1.Width / 2
    frmForm1.Height = frmForm1.Height / 2
    frmForm1.Caption = "Сделай это еще раз!"
End Sub

```

Также форма обладает набором методов и инструкций. **Метод** определяет действие, которое может быть выполнено с объектом. **Инструкция** иницирует действие. Она может выполнить метод или функцию, В табл. 3.3 и 3.4 приведены наиболее часто используемые методы и инструкции для работы формами.

Таблица 3.3. Основные методы форм

Метод	Описание
<i>Hide</i>	Скрывает объект UserForm, но не выгружает его.
<i>Show</i>	Выводит на экран объект UserForm.

Таблица 3.4. Основные инструкции форм

Инструкция	Описание
<i>Load</i>	Загружает объект UserForm, но не отображает его на экране.
<i>Unload</i>	Удаляет объект UserForm из памяти.

В следующем примере предполагается, что в проекте созданы две формы frmForms. При запуске проекта происходит событие Initialize для формы frmForm1, форма frmForm2 загружается и выводится на экран. Когда при помощи мыши выбирается frmForm2, она делается невидимой, и появляется форма frmForm1. Если же выбирается frmForm1, frmForm2 появляется вновь.

'Событие Initialize формы frmForm1.

```

Private Sub UserForm_Initialize()
    Load frmForm2
    frmForm2.Show
End Sub

```

' Событие Click для формы frmForm2

```

Private Sub UserForm_Click()
    frmForm2.Hide
End Sub

```

' Событие Click для формы frmForm1

```

Private Sub UserForm_Click()
    frmForm2.Show
End Sub

```

3.2. Выбор и использование управляющих элементов

Создание управляющих элементов на форме выполняется с помощью **Панели инструментов**, которая выводится на экран командой **Вид → Панель элементов** (рис. 3.2).

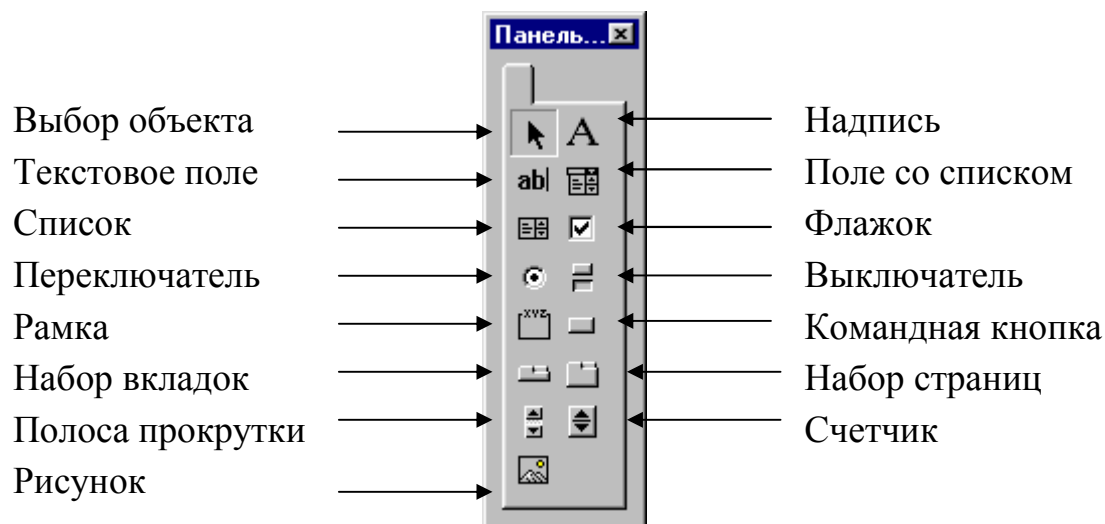


Рис. 3.2. Панель элементов VBA

С помощью кнопок этой панели можно поместить в форму необходимый элемент управления. Для этого нужно щелкнуть на значке элемента управления, далее при нажатой левой кнопке мыши определить размер и место расположения элемента в форме. Когда элемент на форме выделен (рамка объекта содержит маленькие прямоугольники) можно изменять его размеры и перемещать с помощью мыши, а также просматривать и изменять его свойства в окне свойств.

Каждый управляющий элемент (объект) характеризуется набором свойств (которые можно изменять в режимах конструирования или выполнения), событий и методов.

Для каждого объекта проекта необходимо определить его имя. В соответствии с общепринятыми соглашениями об именах объектов первые три символа имени должны отражать вид элемента, а остальные символы - назначение. В табл. 3.5. представлены сочетания первых трех символов для наиболее часто используемых элементов.

Таблица 3.5. Рекомендуемые сочетания первых трех символов имен

Объект	Первые 3 символа имени	Пример имени
Форма	frm	frmMyForm
Надпись	lbl	lblInfo
Текстовое поле	txt	txtInput
Командная кнопка	cmd	cmdExit
Флажок	chk	chkSound
Переключатель	opt	optLevel
Список	lsb	lsbTypes
Рамка	fra	fraChoices
Полоса прокрутки	vcb	vcbSpeed
Рисунок	pic	picChema

Командная кнопка является самым распространенным элементом управления, и может использоваться для организации выполнения вычислений и других действий, вызова процедур и функций пользователя, открытия форм и т.д. Основные свойства командной кнопки представлены в табл. 3.6. В свойстве *Caption* можно ставить символ & перед буквой, которая будет использоваться в сочетании с клавишей Alt для ускоренного доступа к кнопке. Также можно перейти к кнопке клавишей Tab, а затем нажать Enter.

Таблица 3.6. Свойства командных кнопок

Свойство	Описание
<i>BackColor</i>	Цвет фона кнопки.
<i>Caption</i>	Текст, который выводится на кнопке.
<i>Enabled</i>	Значение False делает кнопку недоступной.
<i>Font</i>	Определяет тип и вид шрифта на кнопке.
<i>ForeColor</i>	Определяет цвет шрифта на кнопке.
<i>(Name)</i>	Имя объекта, для программы VBA.
<i>Picture</i>	Добавляет рисунок на кнопку.
<i>PicturePosition</i>	Определяет расположение текста и рисунка на кнопке.
<i>Visible</i>	Значение False делает кнопку невидимой.

Основным **событием** кнопки является **Click**. Для написания программного кода, который будет выполняться при нажатии командной кнопки, достаточно два раза щелкнуть на ней левой кнопкой мыши в режиме конструирования проекта.

Наиболее полезным **методом** командной кнопки является **SetFocus**, позволяющий вернуться к кнопке (передать ей фокус). Например, следующая команда позволяет вернуться к кнопке по умолчанию после ввода данных в текстовое поле: **cmdMyButtum.SetFocus**

Текстовое поле применяется для ввода или вывода информации. Основные свойства текстового поля представлены в табл. 3.7.

Таблица 3.7. Свойства текстового поля

Свойство	Описание
<i>Enabled</i>	Значение False делает поле недоступным.
<i>Font</i>	Определяет тип и вид шрифта в текстовом поле.
<i>ForeColor</i>	Определяет цвет шрифта в текстовом поле.
<i>(Name)</i>	Имя объекта, для программы VBA.
<i>MaxLength</i>	Определяет количество вводимых символов в текстовое поле.
<i>PasswordChar</i>	Определяет символ, отображаемый при вводе в текстовое поле.
<i>Text</i>	Определяет содержимое текстового поля.

Например, для очистки содержимого текстового поля в ходе выполнения программы необходимо ввести в требуемом месте программного кода команду:

```
txtResult.Text=" "
```

Основным **событием** текстового поля является **Change**, происходящее при вводе или удалении символов. Например, команду **cmdMyButtum.SetFocus** можно поместить в процедуру события **Change** текстового поля.

Надпись применяется как самостоятельно для вывода справочной информации, так и в виде "подсказок" для текстового поля, списка или другого элемента. Главное её отличие от текстового поля в том, пользователь не может изменить текст надписи (хотя его можно изменить как свойство во время выполнения программы). Основные свойства надписи представлены в табл. 3.8.

Таблица 3.8. Свойства надписи

Свойство	Описание
<i>Caption</i>	Определяет текст, содержащийся в надписи.
<i>Font</i>	Определяет тип и вид шрифта надписи.
<i>ForeColor</i>	Определяет цвет шрифта надписи.
<i>(Name)</i>	Имя объекта, для программы VBA.
<i>Picture</i>	Добавляет рисунок в надпись.
<i>PicturePosition</i>	Определяет расположение текста и рисунка надписи.

Список позволяет работать с перечнем из нескольких вариантов. Пользователь может просмотреть содержимое списка и выбрать один из вариантов для последующей обработки. Прямое редактирование содержимого списка невозможно. Если в списке помещаются не все строки, то автоматически добавляется вертикальная полоса прокрутки. Основные свойства списка представлены в табл. 3.9.

Таблица 3.9. Свойства списка

Свойство	Описание
<i>(Name)</i>	Имя объекта, для программы VBA.
<i>ListIndex</i>	Возвращает номер текущей выделенной строки списка -1.
<i>Text</i>	Содержимое текущей выделенной строки списка.

Для списка чаще всего используются **события Click** и **DbClick** (двойной щелчок левой кнопкой мыши на одной из строк списка). Во втором случае пользователь одновременно выделяет строку и начинает ее обработку.

Работа со списком начинается с его заполнения **методом AddItem**, который может вызываться несколько раз подряд. Часто метод AddItem помещается в процедуру **UserForm_Initialize()**, чтобы список заполнялся при загрузке формы. Метод **RemoveItem** удаляет строки из списка. Метод **Clear** очищает сразу весь список. Следующий пример показывает, как работают списки, при этом предполагается, что в проекте создана форма с двумя списками (List1 и List2). Двойной щелчок на любой строке одного списка перемещает её в другой список. Строка включается в другой список до того, как она будет удалена из текущего.


```

Private Sub UserForm_Initialize()
    List1.AddItem "Стол"
    List1.AddItem "Стул"
    List1.AddItem "Диван"
    List1.AddItem "Кресло"
    List1.AddItem "Кровать"
End Sub

```

```

Private Sub List1_DblClick()
    List2.AddItem List1.Text
    List1.RemoveItem
    List1.ListIndex
End Sub
Private Sub List2_dblClick()
    List1.AddItem List2.Text
    List2.RemoveItem
    List2.ListIndex
End Sub

```

Переключатели позволяют выбрать один вариант из группы. Обычно они группируются в рамках (см. далее), однако их можно располагать прямо на форме, если используется только одна группа переключателей. Основные свойства переключателя представлены в табл. 3.10.

Таблица 3.10. Свойства переключателя

Свойство	Описание
<i>Caption</i>	Задаёт текст, определяющий назначение переключателя.
<i>(Name)</i>	Имя объекта, для программы VBA.
<i>Value</i>	Значение True указывает, что переключатель выбран.

Наиболее важным является свойство Value значение True (переключатель находится в установленном состоянии), которого в режиме конструирования задается только у одного переключателя в группе. В режиме выполнения это свойство чаще всего проверяется в процедуре события Click кнопки, нажатой после установки нужного переключателя, что позволяет проверить перед вызовом следующей процедуры некоторое условие. Однако определенные действия можно выполнять сразу же после выбора переключателя в процедуре его события **Click**.

Флажок частично аналогичен переключателю, но в отличие от него может использоваться как отдельный самостоятельный элемент. Даже объединенные в группу флажки работают независимо друг от друга. Основные свойства флажков такие же, как и у переключателя (см. табл. 3.10). Однако свойство Value может принимать три значения (флажок нахо-

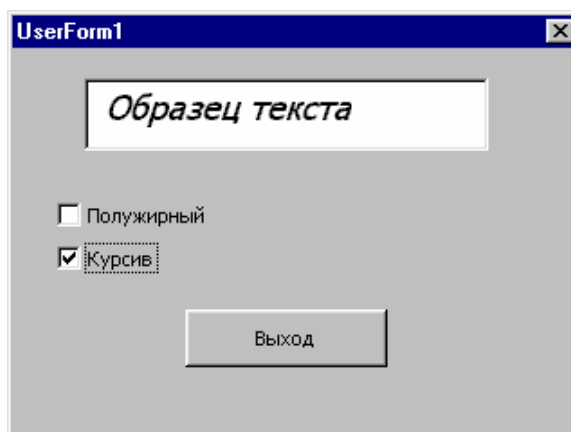


Рис. 3.3. Использование флажков

дится в установленном состоянии, снятом или неопределенном).

Наиболее часто используемым событием флажков является **Click**, в процедуре которого можно проверять состояние флажка по свойству **Value**. Следующий пример иллюстрирует работу флажков, при этом предполагается, что в проекте создана форма с двумя флажками (**ChkBold** и **ChkItalic**) и текстовым полем **TxtExam** (рис. 3.3). После ввода символов в текстовое поле, с помощью флажков можно делать текст полужирным или курсивом. Свойства **FontBold** и **FontItalic** текстового поля устанавливаются способами начертания текста.

```
Private Sub Chkbold_Click()
  If ChkBold.Value = True Then
    TxtExam.FontBold = True
  Else
    TxtExam.FontBold = False
  End If
End Sub
```

```
Private Sub ChkItalic_Click()
  If ChkBold.Value = True Then
    TxtExam.FontItalic = True
  Else
    TxtExam.FontItalic = False
  End If
End Sub
```

Рамка используется для группировки переключателей или флажков, и помещается на форму раньше элементов, находящихся внутри неё. Переключатели находящиеся внутри рамки, работают как самостоятельная группа и не влияют на состояние переключателей в других рамках. Основным свойством рамки является **Caption**, которое задает текст, определяющий назначение элементов в рамке.

Рисунок используется для простейшего вывода изображения на форме. Он может отображать растровые файлы (.BMP), значки (.ICO), метафайлы (WMF), а также файлы в формате JPEG (.JPG) и GIF (.GIF). Основные свойства рисунка представлены в табл. 3.11.

Таблица 3.11. Свойства рисунка

Свойство	Описание
<i>Autosize</i>	Значение True подгоняет размер элемента под размер содержимого.
<i>(Name)</i>	Имя объекта, для программы VBA.
<i>Picture</i>	Задаёт файл для рисунка.

События и методы рамок и рисунков практически не используются.

4. СТРУКТУРА ПРОГРАММЫ. МОДУЛИ, ПРОЦЕДУРЫ И ФУНКЦИИ

Модуль представляет собой текстовый ASCII-файл с программным кодом, содержащим подпрограммы, переменные и константы. Проект может состоять из множества программных модулей. Для их создания необходимо выполнить команду **Вставка** → **Модуль**. Рабочее окно модуля представлено на рис. 4.1.

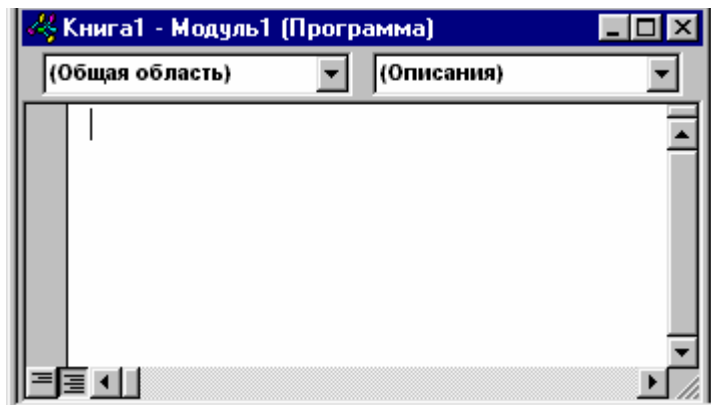


Рис. 4.1. Окно модуля

Основу программ в VBA составляют процедуры и функции.

Процедура Sub— это обособленная совокупность операторов VBA, выполняющая определенные действия. В общем случае процедура принимает некоторые параметры (переменные, которые передаются процедуре в качестве исходных данных), выполняет программу и может возвращать результирующие значения, которые присваиваются параметрам внутри процедуры. Однако чаще используются процедуры без параметров. Например, процедуры, выполняющиеся при возникновении определенных событий. Вложенность процедур в другие процедуры не допускается. Структура процедуры следующая:

```
[ДОСТУП] Sub ИМЯ_ПРОЦЕДУРЫ ([СПИСОК_ПАРАМЕТРОВ])
    ТЕЛО_ПРОЦЕДУРЫ
```

End Sub

Ключевое слово *ДОСТУП* является необязательным и определяет область видимости процедуры. **Public** указывает, что процедура доступна для всех других процедур во всех модулях (глобальная). **Private** указывает, что процедура доступна для других процедур только того модуля, в котором она описана (локальная). *СПИСОК_ПАРАМЕТРОВ* также является необязательным элементом и позволяет передавать процедуре различные исходные данные при вызове, которые называются формальными параметрами. При этом ключевое слово **Dim** не указывается. *ТЕЛО_ПРОЦЕДУРЫ* состоит из описательной части и блока операторов, выполняющихся один за другим. Если необходимо прекратить выполнение процедуры в некотором конкретном месте, это можно сделать с помощью оператора **Exit Sub**. *ИМЯ_ПРОЦЕДУРЫ* – это любой идентификатор, определенный пользователем. **Идентификатор** – это последовательность букв,

цифр и символа подчеркивания, начинающаяся с буквы (пробелы внутри идентификатора недопустимы). Имя процедуры всегда определяется на уровне модуля. Для использования процедуры в тексте программы (т.е. для её вызова), необходимо указать имя процедуры и список фактических параметров, которые должны по типу и порядку следования совпадать с формальными параметрами.

Функция Function во многом похожа на процедуру, но в отличие от неё при вызове всегда возвращает значение. Функция получает параметры, называемые **аргументами**, и выполняет с ними некоторые действия, результат которых возвращается функцией. Структура функции следующая:

```
[ДОСТУП] Function ИМЯ_ФУНКЦИИ(СПИСОК_АРГУМЕНТОВ) As ТИП
    ТЕЛО_ ФУНКЦИИ
ИМЯ_ФУНКЦИИ = ВЫРАЖЕНИЕ
```

End Function

ТИП определяет тип данных возвращаемого результата. В теле функции обязательно должен присутствовать, по крайней мере, один оператор, присваивающий имени функции значение вычисляемого выражения. Досрочное завершение функции возможно с помощью оператора **Exit Function**. В программе вызов функции осуществляется с помощью оператора присваивания, в правой части которого указывается имя функции с перечнем фактических параметров, как и любой другой встроенной функции, например, Sqr, Cos или Chr.

Процедуры и функции, не описанные явно с помощью ключевых слов Public или Private, по умолчанию являются общими.

Для быстрого добавления в модуль подпрограмм удобно воспользоваться командой **Вставка → Процедура**. В появившемся окне (рис. 4.2) нужно выбрать необходимые опции.

В MS Excel с функциями, созданными пользователем, можно работать с помощью Мастера функций точно так же, как и со встроенными функциями рабочего листа.

Пример. В MS Excel создать функцию пользователя, математически определенную следующим образом:

$$y = \sin(x) \cdot e^{-5x}$$

Создадим модуль, как указано выше (рис 4.1), и введем в него текст следующей программы:

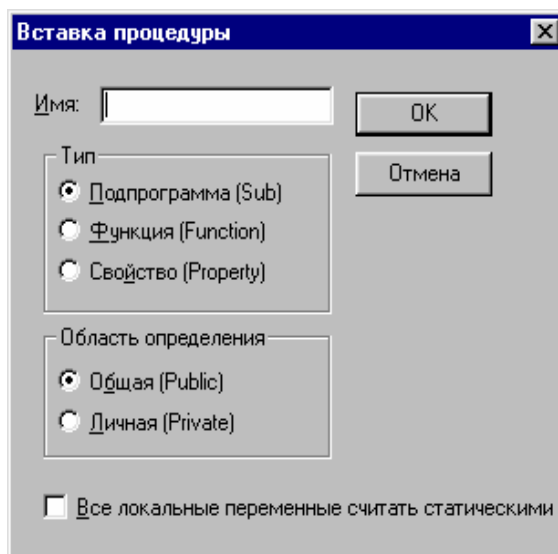


Рис. 4.2. Диалоговое окно “Вставка процедуры”

Public Function Y(x As Single) As Single

$$Y = \text{Sin}(x) * \text{Exp}(- 5 * x)$$

End Function

Для использования созданной функции на рабочем листе MS Excel введем в ячейку A2 число 0.1. В ячейке B2 вычислим значение функции Y при $x = 0.1$. Для этого в ячейку B2 достаточно ввести формулу `=Y(A2)`. Это можно сделать и с помощью Мастера функций, который будет содержать функцию Y наряду с другими встроенными функциями MS Excel (рис 4.3).

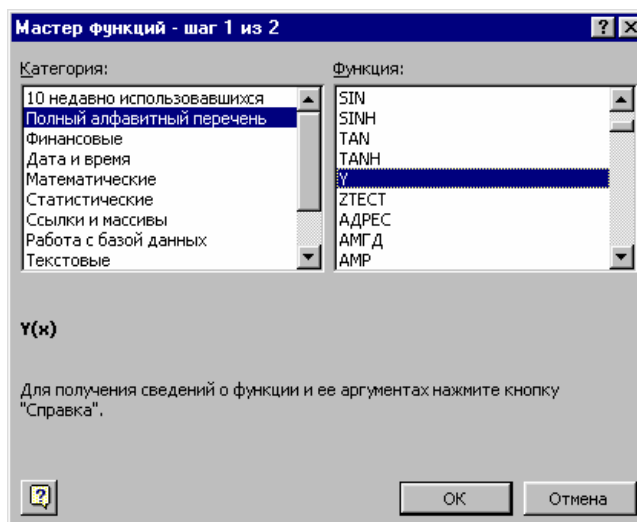


Рис. 4.3. Вызов функции Y()

Общие принципы организации программ VBA в модуле следующие. Обычно текст программы начинается с опций, которые управляют описанием переменных, способом сравнения строк и т. д. Затем следует объявление **глобальных** для данного модуля переменных и констант, т.е. таких, которые используются во всех процедурах модуля. Далее располагают непосредственно текст функций и процедур, составляющих саму программу. Разделителем операторов в одной строке при записи программы является символ “:”. Для переноса оператора на другую строку используется символ “_” (знак подчеркивания).

Иногда внутри программы удобно помещать **комментарии** – пояснительный текст, который игнорируется компилятором и может быть записан в любом месте программы. Комментарии удобно также использовать при отладке программы для временного отключения операторов. Каждая строка комментариев начинается со знака апострофа.

Пример организации модуля:

Option Base 1

Option Explicit

' PI – глобальная константа

Const PI As Double = 3.14159

' x - глобальная переменная

Dim x As Double

' Функция Disc вычисляет площадь круга

Public Function Disc(R As Double) As Double

x = 2

Disc = PI * R ^ 2

End Function

‘ Функция Rec вычисляет площадь треугольника

Public Function Rec(a As Double, b As Double, c As Double) As Double

‘ p – локальная переменная

Dim p As Double

p = (a + b + c) / 2

Rec = Sqr (p * (p – a) * (p – b) * (p – c))

End Function

‘ Процедура Result вызывает функции Disc и Rec и выводит результаты

Public Sub Result ()

‘ R_1, R_2, a, b, c - локальные переменные

Dim R_1 As Double, R_2 As Double, a As Double, b As Double, c As Double

R_1 = Disc(2.5)

x = x + 2

MsgBox “ Площадь круга = “ & CStr(R_1) & “, x = “ & CStr(x)

a = 1: b = 1 : c = Sqr(2)

R_2 = Rec (a, b, c)

MsgBox “ Площадь треугольника = “ & CStr(R_2)

End Sub

Здесь инструкция **Option Explicit** указывает на необходимость описания типов переменных, используемых на данном листе модуля. Инструкция **Option Base 1** указывает, что индексация элементов массива будет начинаться с 1. Функция **Sqr** вычисляет квадратный корень аргумента. Функция **CStr** переводит числовой формат в строковый.

5. ОСНОВНЫЕ ОПЕРАТОРЫ ЯЗЫКА VBA

5.1. Операторы описания

Все значения, с которыми работает программа, хранятся в переменных, константах, массивах и т.п. **Переменной** называется именованная область памяти, в которой могут храниться различные значения. Каждая переменная в VBA имеет свой **тип**, который указывает, что может хранить переменная: целое число, строку, дату и т.д.

В процессе выполнения программы значения переменных могут изменяться с помощью оператора присваивания: “=”. Таким образом, различают **описание** переменных (определение их типа) и **инициализацию** переменных (присвоение им начальных значений). Переменную в VBA можно описать с помощью следующей конструкции:

Dim ИМЯ_ПЕРЕМЕННОЙ **As** ТИП_ПЕРЕМЕННОЙ

Базовые типы данных VBA приведены в табл. 5.1.

Таблица 5.1. Стандартные типы данных

<i>Типы данных</i>	Размер памяти	Назначение
Boolean (Логический)	2	Содержит значения True (Истина) или False (Ложь).
Byte (Положительное целое)	1	Содержит целые положительные числа в диапазоне от 0 до 255.
Integer (Целое)	2	Содержит целые числа в диапазоне от -32768 до 32768.
Long (Длинное целое)	4	Содержит целые числа в диапазоне от -2147483648 до 2147483647.
Single (Число с плавающей точкой)	4	Содержит вещественные числа по абсолютной величине от 1,401298 E-45 до 3,402823 E+38.
Double (Число с плавающей точкой двойной точности)	8	Содержит вещественные числа по абсолютной величине от 4,940656458 E-324 до 1,79769313 E+308.
Date (Дата)	8	Содержит даты от 1.01.100 г. до 1.12. 9999 г.
Object (Объект)	4	Содержит ссылки на любой заданный объект.

String (Строковый)	10 +длина	Содержит текстовые или строковые значения длиной от 0 до $2 \cdot 10^9$.
Currency (Денежный)	8	Содержит денежные величины в диапазоне от -922337203685477,58 до 922337203685477,58.
Variant (Вариант)	Универсальный тип общего назначения, который может хранить значения большинства других типов.	

VBA позволяет не описывать переменные перед их использованием в программе. В этом случае по умолчанию используется тип **Variant**. Переменные этого типа могут хранить все, что в них поместят, т.е. их тип изменяется в зависимости от последнего присвоения. Однако рекомендуется все переменные, которые будут использоваться в программе, описывать явно. Для запрета использования переменных, которые не были явно описаны в начало программы необходимо вставить строку **Option Explicit**.

Например:

- Dim A As Integer** - объявляется переменная A целого типа;
- Dim E As Single** - объявляется переменная E действительного типа;
- Dim S As String** - объявляется строковая переменная S.

Допускается объявлять в одной строке несколько переменных. Например:

Dim B1 As Single, B2 As Single, B3 As Single
Dim B1, B2, B3 As Single

В первом случае все три переменные B1, B2 и B3 будут иметь тип Single. Во втором случае тип Single будет иметь только переменная B3, а переменным B1 и B2 будет присвоен тип Variant. Такое “не существенное” на первый взгляд отличие в объявлении переменных в дальнейшем может привести к серьезным ошибкам в процессе работы программы.

Часто при написании программ необходимо использовать одни и те же постоянные значения: числа, строки, даты и т.д. В этом случае можно задать константу с помощью одной из следующих конструкций:

Const ИМЯ_ПОСТОЯННОЙ = ВЫРАЖЕНИЕ
Const ИМЯ_ПОСТОЯННОЙ As ТИП_ПОСТОЯННОЙ = ВЫРАЖЕНИЕ

Например:

Const FileName = "text.xls"
Const PI As Double = 3.14159

Для хранения векторов, матриц и т.д. можно использовать массивы. **Массив** - это структурированный тип данных, который представляет собой последовательность ячеек памяти, имеющих общее имя и хранящих данные одного типа. Каждый элемент массива определяется индексом (номером). Количество элементов в массиве называется **размерностью массива**. Массив описывается следующей конструкцией:

Dim ИМЯ_МАССИВА (Размерность) **As** ТИП_МАССИВА

Например:

Dim A(12) As Integer – объявляется вектор из 12 целых чисел, причем по умолчанию первый элемент массива будет A(0), а последний A(11). Говорят, что 0 – **базовый индекс**.

Dim B(3, 3) As Single – объявляется матрица 3 x 3 действительных чисел.

Чтобы задать массив, у которого 1-й элемент будет иметь индекс 1, необходимо использовать следующую конструкцию:

Dim ИМЯ_МАССИВА (1 to Размерность) **As** ТИП_МАССИВА

Например:

Dim B(1 to 3, 1 to 3) As Single - использование ключевого слова **to** при объявлении массива изменяет базовый индекс на 1.

Dim A(1 to 12) As Integer

Массив в программе инициализируется поэлементно, например:

Dim B (1 to 2) As Single

B(1) = 2.1 : B(2) = 4.6

Иногда размер массива не может быть определен заранее. В таком случае можно объявить массив без указания размерности, который называется **динамическим**.

Dim ИМЯ_МАССИВА() **As** ТИП_МАССИВА

Например: **Dim Y() As Integer**

Затем добавление элементов в массив осуществляется оператором **ReDim**, который можно использовать только в процедурах.

Например: **ReDim Y(5)**

Созданным элементам необходимо присвоить значения. Позднее количество элементов массива снова можно изменить. При этом все значения, хранящиеся в массиве, теряются. Чтобы сохранить их необходимо использовать ключевое слово **Preserve**.

Например: **ReDim Preserve Y(7)**

5.2. Оператор присвоения имеет следующий синтаксис:

ИМЯ_ПЕРЕМЕННОЙ = ВЫРАЖЕНИЕ

ВЫРАЖЕНИЕ - это любые арифметические и логические операторы, операторы сравнения и конкатенации, написанные по правилам VBA. Результат вычисления выражения заносится в указанную переменную. *ИМЯ_ПЕРЕМЕННОЙ* или **идентификатор** – не может начинаться с цифры и содержать следующие символы: “.”, “;”, “>”, “<”, “=”, арифметические знаки. В таблице 5.2 приведены арифметические операторы.

Таблица 5.2. Арифметические операторы

Оператор	Описание
Оператор ^	Возводит число в степень
Оператор *	Возвращает произведение двух чисел
Оператор /	Возвращает результат деления двух чисел
Оператор \	Возвращает результат целого деления двух чисел
Оператор Mod	Возвращает остаток при целом делении двух чисел (значение по модулю)
Оператор +	Возвращает сумму двух чисел
Оператор -	Возвращает разность двух чисел или изменяет знак числового выражения

Операторы приведены в порядке убывания их приоритета.

Для избежания ошибок рекомендуется выполнять присвоение только между совместимыми типами данных. Например, переменная целого типа не может принимать строковое значение, если вся строка не распознается как целое число. Тип Integer всегда может быть присвоен типу Long, тип Single всегда может быть присвоен типу Double, и любой тип (кроме типа, определяемого пользователем) может быть присвоен типу Variant. При присвоении вещественного типа целому происходит округление (если результат округления не превышает максимального значения, которое может храниться в этом типе).

Если выражение содержит несколько операций, то значения компонентов выражения рассчитываются в определенном порядке, который называют приоритетом операций.

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

Стоящие рядом в выражении операции умножения и деления выполняются слева направо. В таком же порядке выполняются стоящие рядом операции сложения и вычитания. Операции внутри круглых скобок всегда выполняются раньше, чем операции вне скобок. Порядок выполнения операций, стоящих внутри скобок, определяется их приоритетом.

Например, математическое выражение $y = \frac{x^2 - 3 \cdot x^{0.7}}{(0.2 \cdot x + 1) \cdot (x - 4)}$ на VBA

будет иметь следующий вид:

$$y = (x ^ 2 - 3 * x ^ (0.7)) / ((0.2 * x + 1) * (x - 4))$$

Правая часть оператора присваивания может содержать стандартные функции VBA. В таблицах 5.3 и 5.4 приведены наиболее часто используемые стандартные функции преобразования типов и математические функции VBA.

Таблица 5.3. Функции преобразования типов

Функция	Описание
CDate(выражение)	Преобразует выражение в тип Date.
CInt(выражение)	Преобразует выражение в тип Integer.
CLng(выражение)	Преобразует выражение в тип Long.
CSng(выражение)	Преобразует выражение в тип Single.
CStr(выражение)	Преобразует выражение в тип String.

Таблица 5.4. Математические функции

Функция	Описание
abs(аргумент)	Возвращает значение, тип которого совпадает с типом переданного аргумента, равное абсолютному значению указанного числа.
atn(аргумент)	Возвращает значение типа Double, содержащее арктангенс числа.
cos(аргумент)	Возвращает значение типа Double, содержащее косинус угла.
int(аргумент)	Возвращает значение типа, совпадающего с типом аргумента, которое содержит целую часть числа.
log(аргумент)	Возвращает значение типа Double, содержащее натуральный логарифм числа.
exp(аргумент)	Возвращает значение типа Double, содержащее результат возведения числа e (основание натуральных логарифмов) в указанную степень.
sin(аргумент)	Возвращает значение типа Double, содержащее синус угла.
sqr(аргумент)	Возвращает значение типа Double, содержащее квадратный корень указанного числа.
tan(аргумент)	Возвращает значение типа Double, содержащее тангенс угла.

Например, математическое выражение $y = \sin(\sqrt{|1.2x|}) - e^x + \ln(1+|x|)$ на VBA будет иметь следующий вид: **y = sin(sqrt(abs(1.2*x))) - exp(x) + log(1+abs(x))**

5.3. Операторы ввода-вывода

5.3.1. Функция MsgBox осуществляет вывод информации в диалоговом окне и устанавливает режим ожидания нажатия кнопки пользователем. Она имеет следующий синтаксис:

MsgBox *СООБЩЕНИЕ*[,*КНОПКИ*][, *ЗАГОЛОВОК*]


Аргументы:

СООБЩЕНИЕ - обязательный аргумент, задающий в окне выводимое информационное сообщение. Может состоять из нескольких текстовых строк, объединенных знаком **&**. Использование в этом аргументе **Chr(13)** приводит к переходу на новую строку при выводе информации.

КНОПКИ - значение этого аргумента определяет категории появляющихся в окне кнопок. От значения аргумента кнопки зависят также, появляется ли в окне какой-либо значок. Если не указано, какие кнопки необходимо отображать в окне сообщений, то используется значение по умолчанию, соответствующее кнопке ОК. В табл. 5.1 приведены возможные комбинации кнопок и значков в окне сообщений.

ЗАГОЛОВОК - задает заголовок окна.

Таблица 5.5. Допустимые значения переменной кнопки

Отображение	Аргумент
Кнопка ОК	VbOKOnly
Кнопки ОК и Отмена	VbOKCancel
Кнопки Да и Нет	VbYesNo
Кнопки Да, Нет и Отмена	VbYesNoCancel
Кнопки Прекратить, Повторить и Игнорировать	VbAbortRetryIgnore
Кнопки Повторить и Отмена.	VbRetryCancel
Информационный знак	VbInformation
Знак 	VbCritical
Знак вопроса	VbQuestion
Знак восклицания	VbExclamation

Функция **MsgBox** возвращает значение типа Integer, указывающее, какая кнопка была нажата в диалоговом окне.

Например:

```
MsgBox "Сумма = " & CStr(S) & Chr(13) & "Кол-во = " & CStr(k),_
vbOKOnly, "Результаты"
```

Выводит в отдельное окно с именем "Результаты", содержащее только кнопку "Ok", значение переменных S и k, с соответствующими пояснениями, в две строки (рис. 5.1).

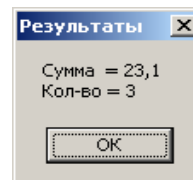


Рис. 5.1. Использование функции MsgBox

Для вывода с помощью одной функции MsgBox элементов одномерного массива (рис. 5.2) можно воспользоваться следующим фрагментом программы:

‘Описание строковой переменной St для хранения выводимой информации

```
Dim St As String
```

‘Формирование элементов массива

...

‘Присваивание начального значения переменной St

```
St = "Массив X" & Chr(13)
```

‘Заполнение переменной St значениями элементов

‘массива X

```
For i = 1 To N
```

```
St = St & "X(" & CStr(i) & ")= " & CStr(x(i)) & Chr(13)
```

```
Next i
```

‘Вывод переменной St, содержащей массив X

```
MsgBox St, vbOKOnly, "Результаты"
```

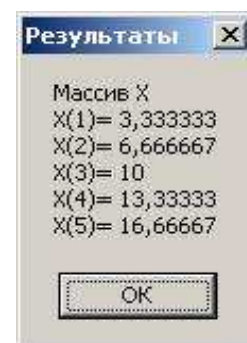


Рис. 5.2. Вывод массива

5.3.2. Функция **InputBox** осуществляет ввод значений переменных с помощью окна ввода и имеет следующий синтаксис:

```
ИМЯ_ПЕРЕМЕННОЙ = InputBox( СООБЩЕНИЕ[, ЗАГОЛОВОК])
```

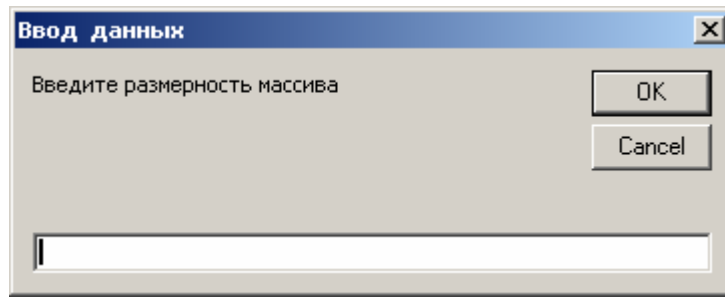
Аргументы:

СООБЩЕНИЕ - обязательный аргумент. Задаёт в окне информационное сообщение, обычно поясняющее смысл вводимой величины

ЗАГОЛОВОК - задаёт заголовок окна.

Например:

```
n = InputBox("Введите размерность массива", "Ввод данных") -
```

Рис. 5.3. Использование функции **InputBox**

5.4. Условные операторы

Для реализации разветвляющегося вычислительного процесса в VBA используется оператор **If...Then...Else**, который представляет собой простейшую форму проверки условий. Он имеет следующий синтаксис:

If УСЛОВИЕ Then ОПЕРАТОР_1 Else ОПЕРАТОР_2

ОПЕРАТОР_1 выполняется, если *УСЛОВИЕ* истинно, в противном случае выполняется *ОПЕРАТОР_2*. При этом оператор If...Then...Else записывается в одну строку.

УСЛОВИЕ – это выражение логического типа. Результат выражения всегда имеет булевский тип. Выражение может быть простым и сложным. При записи простых условий могут использоваться все возможные операции отношения, указанные в табл. 5.6.

Таблица 5.6. Логические отношения

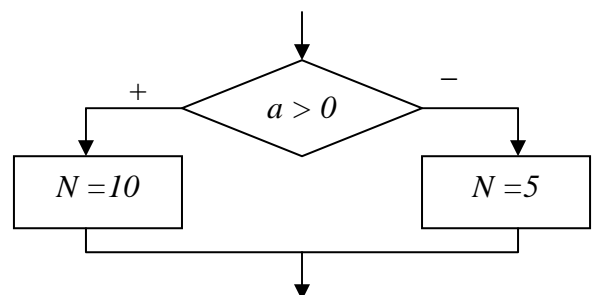
Операция	Название	Выражение	Результат
=	Равно	A = B	True, если A равно B
<>	Не равно	A <> B	True, если A не равно B
>	Больше	A > B	True, если A ,больше B
<	Меньше	A < B	True, если A меньше B
>=	Больше или равно	A >= B	True, если A больше или равно B
<=	Меньше или равно	A <= B	True, если A меньше или равно B

Например, вычислить значение выражения:

$$N = \begin{cases} 10, & \text{если } a > 0 \\ 5, & \text{если } a \leq 0 \end{cases}$$

можно с помощью оператора:

If a > 0 Then N = 10 Else N = 5



Ветвь Else в условном операторе является необязательной частью. В этом случае после Then можно разместить несколько операторов, для того, чтобы все они выполнялись, если условие истинно. При этом они должны располагаться в одну строку и быть разделены двоеточием, например:

If A > 10 Then A = A+1 : B = B + A : C = C * A

Сложные условия образуются из простых путем применения логических операций и круглых скобок. Список логических операций приведен в табл. 5.7.

Таблица 5.7. Логические операции

Операция	Название	Выражение	A	B	Результат
Not	Логическое отрицание	Not A	False		True
			True		False
And	Логическое И	A And B	True	True	True
			True	False	False
			False	True	False
			False	False	False
Or	Логическое ИЛИ	A Or B	True	True	True
			True	False	True
			False	True	True
			False	False	False

В условном операторе допустимо использование блока операторов вместо любого из операторов. В этом случае условный оператор имеет вид:

```
If УСЛОВИЕ Then
    БЛОК_ОПЕРАТОРОВ_1
Else
    БЛОК_ОПЕРАТОРОВ_2
End If
```

В условном операторе может проверяться несколько условий. В этом случае условный оператор имеет вид:

```
If УСЛОВИЕ_1 Then
    БЛОК_ОПЕРАТОРОВ_1
ElseIf УСЛОВИЕ_2 Then
    БЛОК_ОПЕРАТОРОВ_2
Else
    ....
End If
```

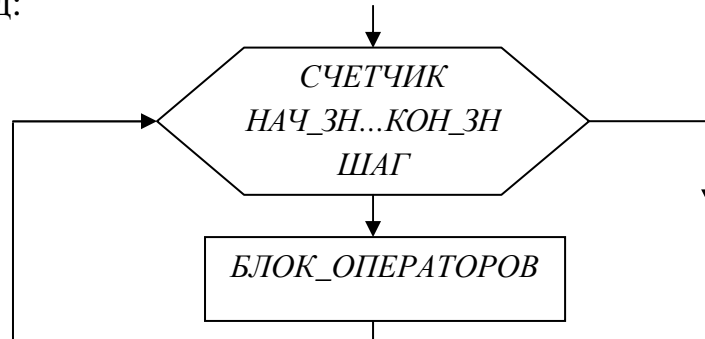
5.5. Операторы цикла

Для реализации циклического вычислительного процесса, т. е. многократного выполнения одного или нескольких операторов, служит оператор цикла **For...Next**, который имеет следующий синтаксис:

```
For СЧЕТЧИК=НАЧ_ЗНАЧЕНИЕ To КОН_ЗНАЧЕНИЕ Step ШАГ
    БЛОК_ОПЕРАТОРОВ
[Exit For]
    БЛОК_ОПЕРАТОРОВ
Next СЧЕТЧИК
```

Цикл **For...Next** перебирает значения переменной *СЧЕТЧИК*, которая является параметром цикла, от начального до конечного значения с указанным шагом изменения. При этом обеспечивается выполнение блока операторов тела цикла при каждом новом значении счетчика. Если **Step ШАГ** в конструкции отсутствует, то по умолчанию считается, что шаг равен 1. По оператору **Exit For** можно выйти из оператора цикла до того, как *СЧЕТЧИК* достигнет последнего значения.*

Фрагмент блок-схемы алгоритма соответствующий оператору цикла **For...Next** имеет вид:



Для перебора объектов из группы подобных объектов, например, ячеек из диапазона или элементов массива, удобно использовать оператор цикла **For...Each...Next**.

```
For Each ЭЛЕМЕНТ In ГРУППА
    БЛОК_ОПЕРАТОРОВ
[Exit For]
    БЛОК_ОПЕРАТОРОВ
Next ЭЛЕМЕНТ
```

* **Примечание.** Не рекомендуется принудительно изменять значения параметра цикла, его начального и конечного значения в теле цикла **For...Next**.

В VBA для организации циклов с неизвестным заранее числом повторений используются и другие операторы цикла:

циклы с предусловием – **Do While ... Loop**, **Do Until ... Loop**;

циклы с постусловием – **Do ... Loop While**, **Do ... Loop Until**.

Ниже приведены синтаксис этих операторов цикла и показан их принцип работы в виде блок-схемы:

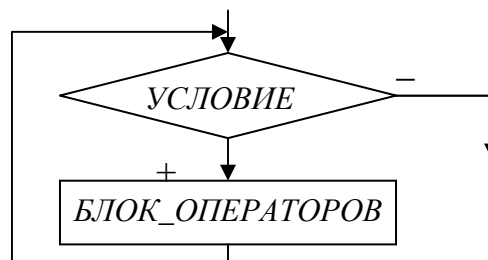
Do While УСЛОВИЕ

БЛОК_ОПЕРАТОРОВ

[Exit Do]

БЛОК_ОПЕРАТОРОВ

Loop



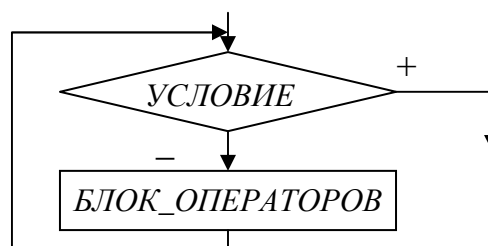
Do Until УСЛОВИЕ

БЛОК_ОПЕРАТОРОВ

[Exit Do]

БЛОК_ОПЕРАТОРОВ

Loop



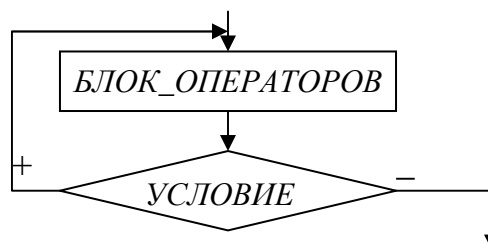
Do

БЛОК_ОПЕРАТОРОВ

[Exit Do]

БЛОК_ОПЕРАТОРОВ

Loop While УСЛОВИЕ



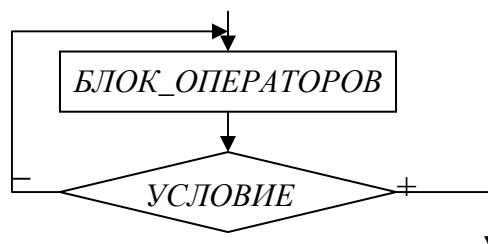
Do

БЛОК_ОПЕРАТОРОВ

[Exit For]

БЛОК_ОПЕРАТОРОВ

Loop Until УСЛОВИЕ



Оператор **Do While...Loop** обеспечивает многократное повторение блока операторов до тех пор, пока *УСЛОВИЕ* соблюдается, а оператор **Do Until...Loop** пока *УСЛОВИЕ* не соблюдается. Операторы **Do...Loop While**, **Do...Loop Until** отличаются от перечисленных выше операторов тем, что сначала блок операторов выполняется по крайней мере один раз, а потом проверяется *УСЛОВИЕ*. Для избежания заикливания в теле цикла должен быть хотя бы один оператор, который изменяет значения переменных, стоящих в *УСЛОВИИ*. Оператор **Exit Do** обеспечивает досрочный выход из оператора цикла.

* * *

В разделе 5 приведен синтаксис наиболее часто используемых операторов VBA, которые необходимо будет использовать при разработке программных модулей. Цель разделов 1-5 – в компактной форме предоставить пользователю справочный материал для обращения к нему ходе практической работы над индивидуальными заданиями.



6. ПРИМЕРЫ РАЗРАБОТКИ ПРОЕКТОВ В СРЕДЕ VBA

6.1. Пример организации разветвляющихся вычислительных процессов

Составить алгоритм и программу для вычисления значения y в соответствии с исходными данными x и a .

$$y = \begin{cases} x^{0.5} + 1.2 \cdot a, & \text{если } -1.5 \leq x \leq 1.5 \\ \sqrt[3]{x-1} - 5.4 \cdot a, & \text{если } 2.5 \leq x \leq 3.5 \\ \sin \frac{\pi}{2} x + \sqrt{a}, & \text{в остальных случаях} \end{cases}$$

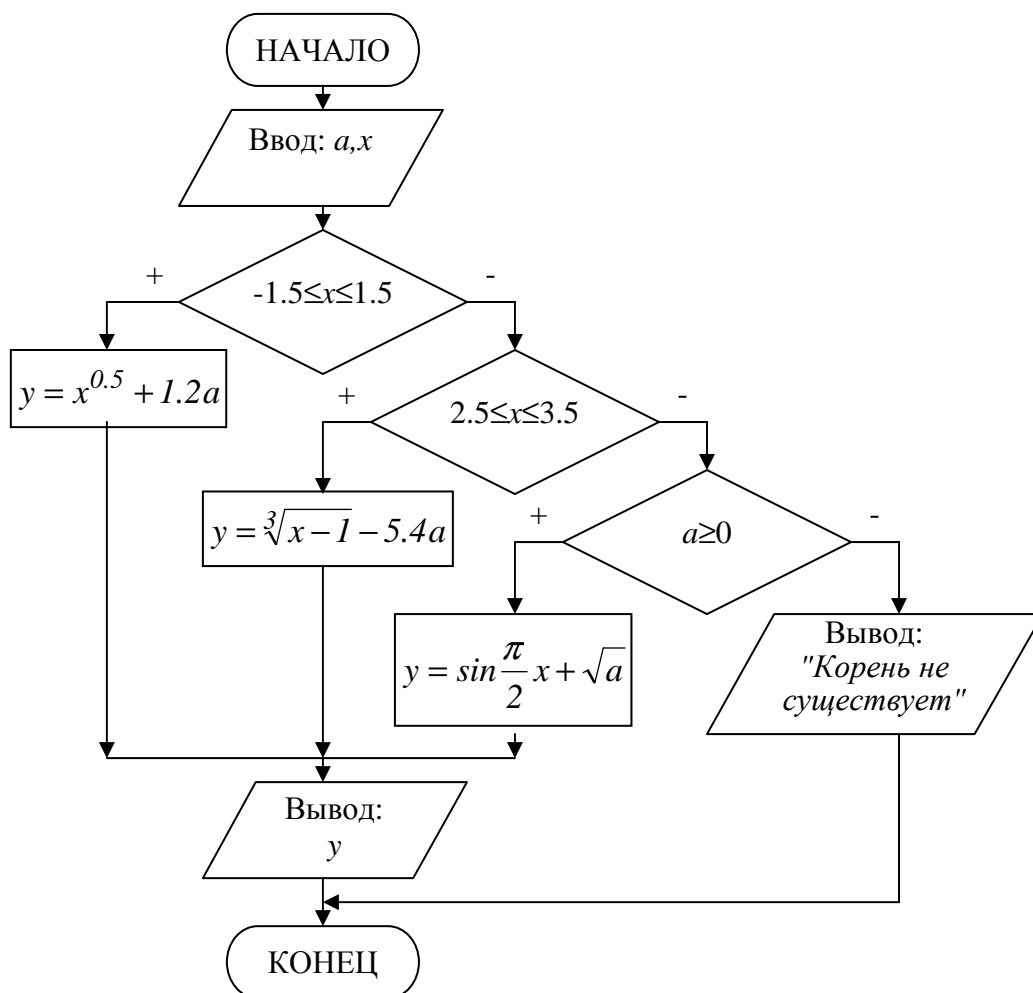
1. Вводимые данные: a, x

2. Мат. модель:

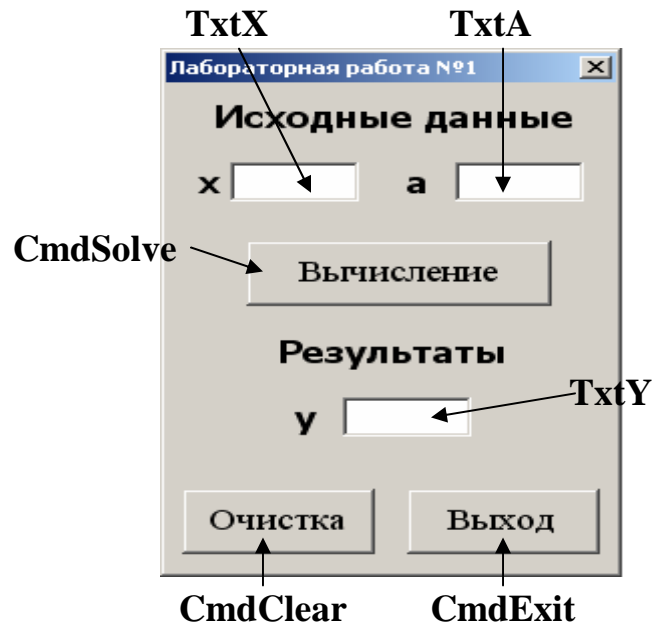
3. Аномалии: $a \geq 0$

4. Выводимые данные: y

5. Блок-схема алгоритма:



Для выполнения задания необходимо создать форму, содержащую поля для ввода исходных данных и вывода результатов, а также командные кнопки для управления работой программы. Для наглядности все управляющие элементы в форме переименованы в соответствии с выполняемой функцией. Ниже приведен текст процедур, выполняющихся при нажатии соответствующих командных кнопок.



```
Const Pi = 3.14
Dim a As Single, x As Single
Dim y As Single
'Расчет требуемых величин при
'нажатии кнопки "Вычисление"
Private Sub CmdSolve_Click()
    a = CSng(TxtA.Text)
    x = CSng(TxtX.Text)
    If x >= -1.5 And x <= 1.5 Then
        y = x ^ 0.5 + 1.2 * a
    ElseIf x >= 2.5 And x <= 3.5 Then
        y = (x - 1) ^ (1 / 3) - 5.4 * a
    ElseIf a >= 0 Then
        y = Sin(Pi / 2 * x) + Sqr(a)
    Else
        MsgBox "Корень не существует", _
            vbOKOnly, "Ошибка!"
```

```
        GoTo m1
    End If
    TxtY.Text = CStr(y)
m1:
End Sub
'Очистка всех текстовых полей
'при нажатии кнопки "Очистка"
Private Sub CmdClear_Click()
    TxtX = "": TxtA = "": TxtY = ""
End Sub
'Завершение работы программы
'при нажатии кнопки "Выход"
Private Sub CmdExit_Click()
    End
End Sub
```

6.2. Пример организации циклов с известным числом повторений

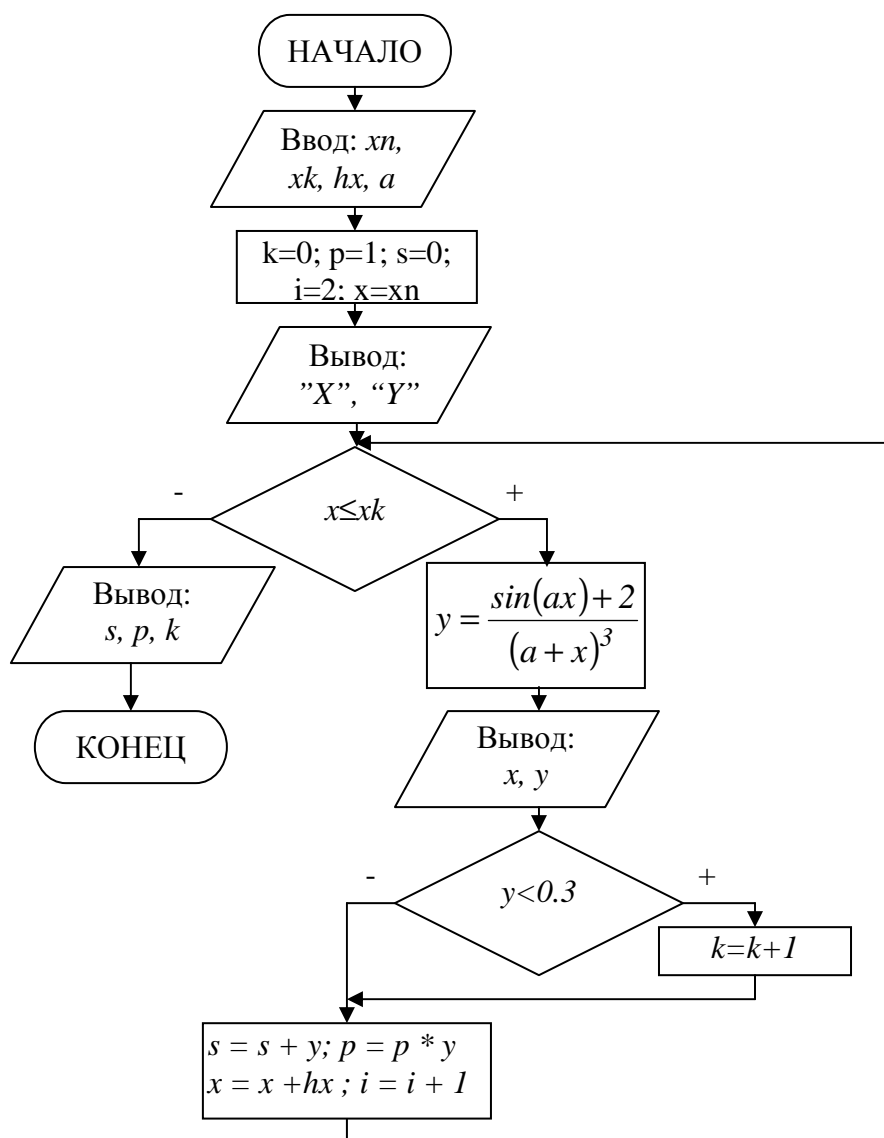
Составить алгоритм и программу для вычисления y , при всех возможных значениях x , которые лежат в интервале $[-3;6]$ с шагом 0.8.

$$y = \frac{\sin(ax) + 2}{(a+x)^3}$$

Вычислить количество $y < 0.3$, сумму и произведение вычисленных y .

1. Вводимые данные: x_n, x_k, h_x, a
2. Мат. модель: $y = \frac{\sin(ax) + 2}{(a+x)^3}$
3. Аномалии: нет
4. Выводимые данные: x, y, k, s, p

5. Блок-схема алгоритма:



Для выполнения задания необходимо создать форму, содержащую поля для ввода исходных данных и вывода итоговых значений, а также командные кнопки для управления работой программы. Для наглядности все управляющие элементы в форме переименованы в соответствии с выполняемой функцией. Все вычисленные значения и соответствующие им значения x выводятся на лист Excel в столбцы A и B. Ниже приведен текст процедур, выполняющихся при нажатии соответствующих командных кнопок:

```
Dim a As Single, x As Single
Dim y As Single, hx As Single
Dim p As Single, s As Single
Dim xn As Single, xk As Single
Dim k As Integer, i As Integer
```

'Очистка всех текстовых полей и листа Excel при нажатии кнопки "Очистка"

```
Private Sub CmdClear_Click()
    TxtA.Text = " "
    TxtXn.Text = " ": TxtXk.Text = " "
    TxthX.Text = " ": TxtP.Text = " "
    TxtS.Text = " ": Txtk.Text = " "
    Sheets(1).Activate
    Range("A1:B50").Clear
End Sub
```

'Завершение работы программы при нажатии кнопки "Выход"

```
Private Sub CmdExit_Click()
    End
End Sub
```

'Расчет требуемых величин при нажатии кнопки "Вычисление"

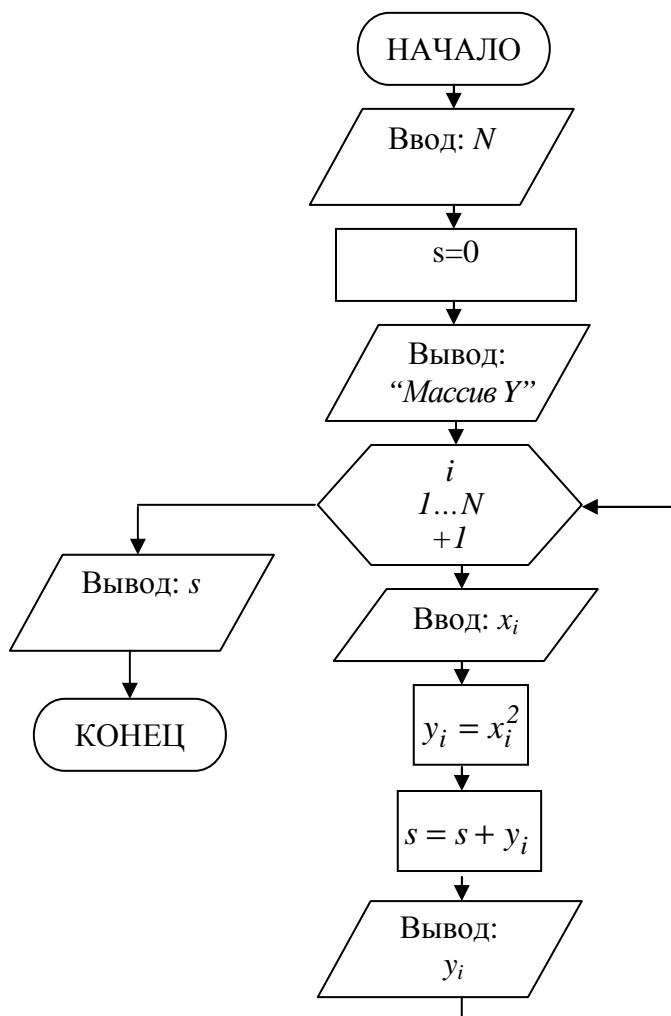
```
Private Sub CmdSolve_Click()
    a = CSng(TxtA.Text)
    xn = CSng(TxtXn.Text)
    xk = CSng(TxtXk.Text)
    hx = CSng(TxthX.Text)
    x=xn: k = 0: p = 1: s = 0: i = 2
    Cells(1, 1) = "X"
    Cells(1, 2) = "Y"
    Do While x ≤ xk
        y = (Sin(a * x) + 2) / (a + x) ^ 3
        Cells(i, 1) = x
        Cells(i, 2) = y
        If y < 0.3 Then k = k + 1
        s = s + y
        p = p * y
        x = x + hx : i = i + 1
    Loop
    Txtk.Text = CStr(k)
    TxtP.Text = CStr(p)
    TxtS.Text = CStr(s)
End Sub
```

6.3. Пример обработки одномерных массивов

На основе элементов исходного массива X , вычислить значения элементов массива Y , по формуле $y_i = x_i^2$. Найти сумму элементов массива Y .

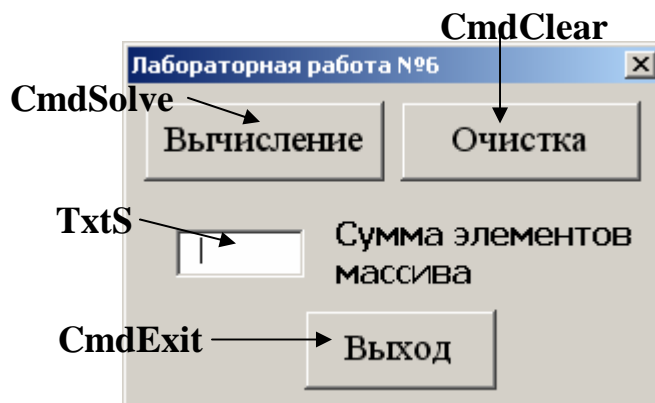
1. Вводимые данные: N , массив X
2. Мат. модель: $y_i = x_i^2$
3. Аномалии: нет
4. Выводимые данные: массив Y , s

5. Блок-схема алгоритма:



Для выполнения задания необходимо создать форму, содержащую командные кнопки и поле для вывода итогового значения.

	A	B	C	D	E	F
1	Кол-во элементов			6		
2						
3	Массив X					
4	2	5	-4	8	1	6
5	Массив Y					
6	4	25	16	64	1	36



Разрядность массива X и значения его элементов, предварительно вводятся на листе MS Excel. Там же будут выводиться и вычисляемые значения элементов массива Y . Ниже приведен текст программы для нажатия управляющих кнопок:

```
Dim x(1 To 10) As Single
```

```
Dim y(1 To 10) As Single
```

```
Dim i, n As Integer
```

```
Dim s As Single
```

'Очистка всех текстовых полей и листа 'Excel при нажатии кнопки "Очистка"

```
Private Sub CmdClear_Click()
```

```
    TxtS.Text = ""
```

```
    Sheets(1).Activate
```

```
    Range("A1:Z6").Clear
```

```
End Sub
```

'Завершение работы программы

'при нажатии кнопки "Выход"

```
Private Sub CmdExit_Click()
```

```
    End
```

```
End Sub
```

'Расчет элементов массива Y и суммы S при нажатии кнопки "Вычисление"

```
Private Sub CmdSolve_Click()
```

```
    n = Cells(1, 4): s = 0
```

```
    Cells(6, 1) = "Массив Y"
```

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

```
        x(i) = Cells(4, i): y(i) = x(i) ^ 2
```

```
        s = s + y(i): Cells(6, i) = y(i)
```

```
    Next i
```

```
    TxtS.Text = CStr(s)
```

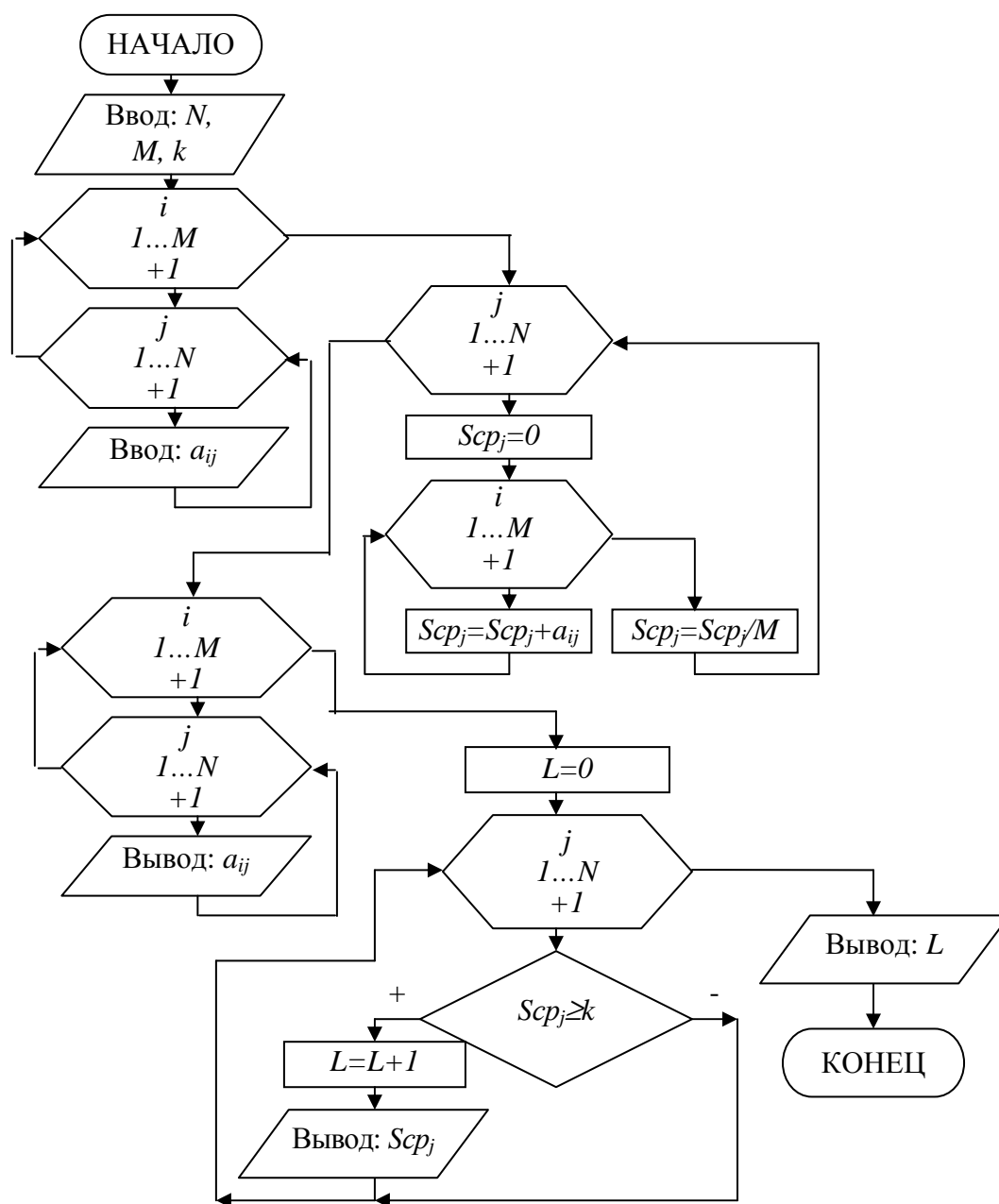
```
End Sub
```


6.4. Пример обработки двумерных массивов

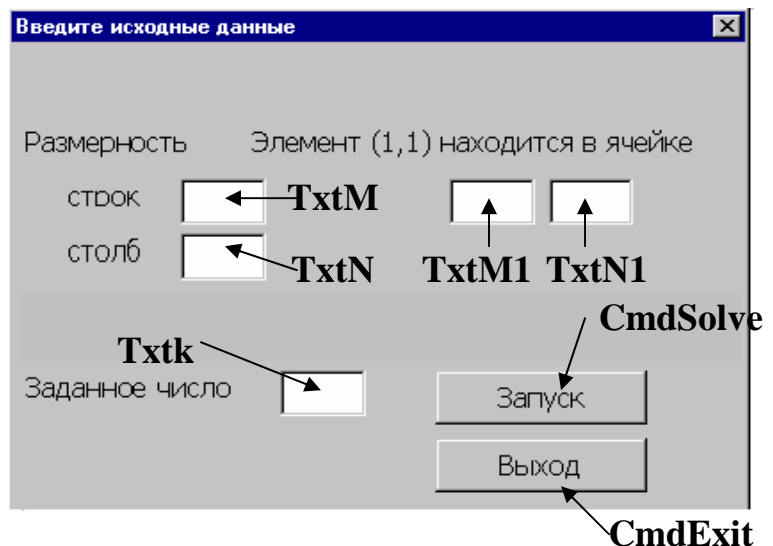
В матрице размерностью 2×9 , для каждого столбца найти среднее арифметическое элементов столбца Scp_j , и вывести те Scp_j , для которых $Scp_j \geq k$, где k - заданная величина. Определить количество таких столбцов.

1. Вводимые данные: k, N, M матрица A
2. Мат. модель: $Scp_j = \frac{\sum_{i=1}^M a_{i,j}}{M}$
3. Аномалии: нет
4. Выводимые данные: L , массив Scp

5. Блок-схема алгоритма:



Для выполнения задания необходимо создать форму, изображенную на рисунке. Текст процедур, выполняемых при нажатии кнопок **Запуск** и **Выход** приведен ниже:



```
Private Sub CmdSolve_Click()
Dim a(10, 10) As Single
Dim sa(10) As Single
Dim i, j, n, m, k, L As Integer
Dim m1, n1 As Integer
' Ввод данных
' Считывание заданного числа
k = CSng(Ttxtk)
' Считывание размера матрицы
m = CInt(TtxtM)
n = CInt(TtxtN)
' Считывание адреса ячейки первого
' элемента
m1 = CInt(TtxtM1)
n1 = CInt(TtxtN1)
' Считывание элементов матрицы
' с "Листа1" MS Excel
Sheets("Лист1").Activate
For i = 1 To m
For j = 1 To n
a(i, j) = Cells(i + m1 - 1, j + n1 - 1)
Next j : Next i
' Расчет
For j = 1 To n
Sa(j) = 0
For i = 1 To m
Sa(j) = sa(j) + a(i, j)
```

```
Next i
Sa(j) = sa(j) / m
Next j
' Вывод матрицы на "Лист2" MS Excel
Sheets("Лист2").Activate
Cells(1, 1) = "Вывод матрицы"
For i = 1 To m
For j = 1 To n
Cells(i + 1, j) = a(i, j)
Next j : Next i
' Вывод результата на "Лист2"
L=0
Cells(n+5, 1) = "Количество="
For j = 1 To n
If sa(j) >= k Then Cells(n + 3, j) = sa(j) :
-
L=L+1
Next j
Cells(n+5, 3) = L
End Sub

Private Sub CmdExit_Click()
End
End Sub
```

Выводимые результаты на втором листе Excel выглядят следующим образом:

	A	B	C	D	E	F	G	H	I
1	Вывод матрицы								
2	4,12	9,04	22,77	13,11	7,09	2,011	6,723	4,17	18,59
3	5,76	0,951	2,125	20,59	3,88	7,932	16,22	1,02	11,22
4									
5			12,4475	16,85			11,4715		14,905
6									
7	Количество=		4						

6.5. Пример обработки табличных данных на рабочем листе MS EXCEL

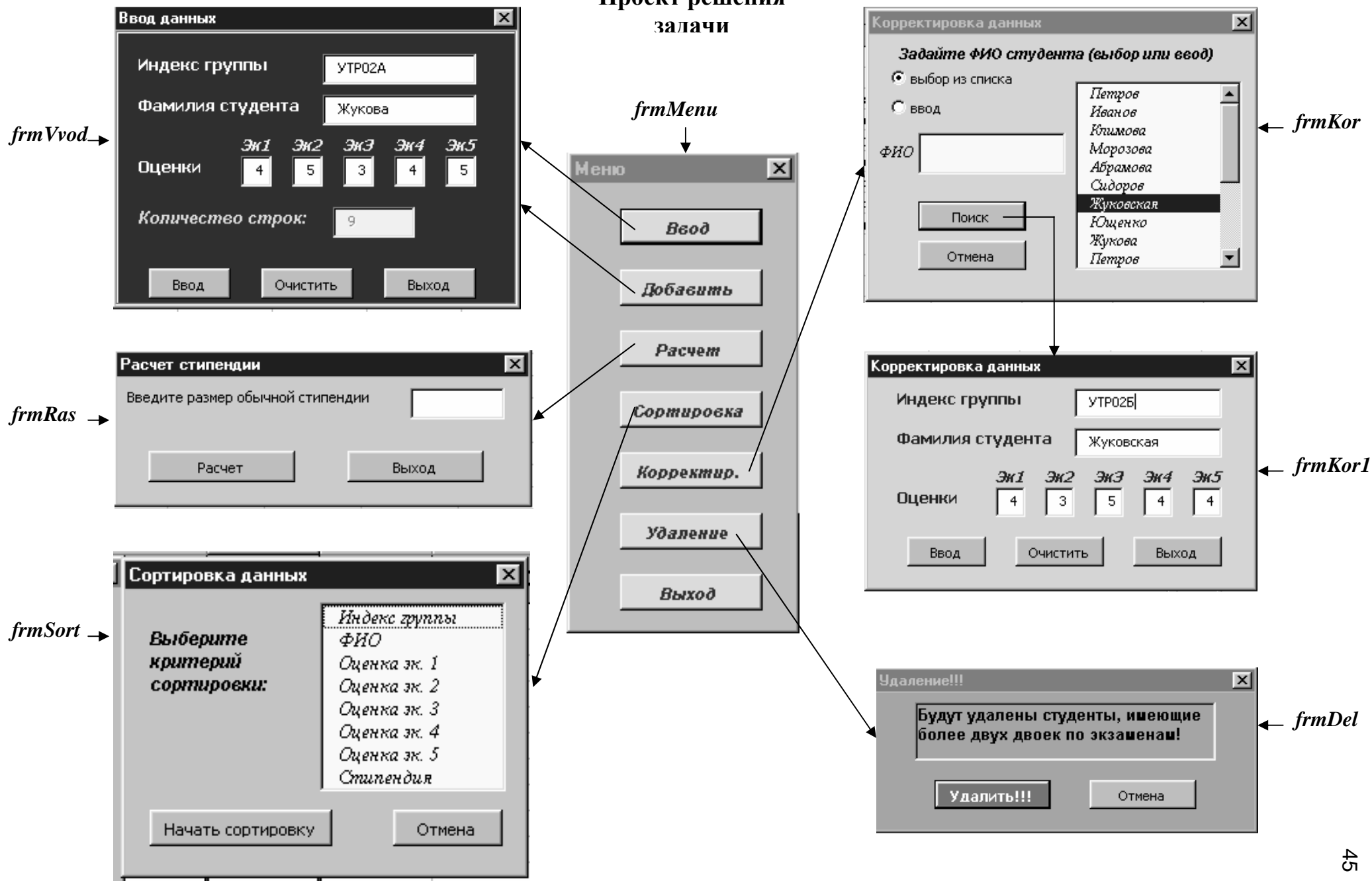
Задание. Создать проект, формы пользователя и программные модули для обработки сессии студентов:

1. Создать таблицу следующей структуры:
 - индекс группы;
 - фамилия студента;
 - оценки по пяти экзаменам;
 - стипендия;
2. Добавить в таблицу произвольное количество строк;
3. Рассчитать стипендию студентам по результатам сессии, причем отличникам стипендию увеличить на 100%, сдавшим без троек – увеличить на 30%, а при наличии хотя бы одной двойки стипендию не начислять;
4. Сортировать данные в таблице по выбранной колонке;
5. Изменить показатели студента с заданной фамилией;
6. Удалить из таблицы данные о студентах, имеющих более двух двоек.

Пример таблицы:

Индекс группы	ФИО студента	Экзамен 1	Экзамен 2	Экзамен 3	Экзамен 4	Экзамен 5	Стипендия
УТР02Б	Петров	5	4	5	4	2	0
УТР02А	Иванов	4	5	3	3	3	100
УТР02А	Климова	5	4	5	4	3	100
УТР02А	Морозова	3	3	3	3	3	100
УТР02А	Абрамова	4	4	5	3	3	100
УТР02А	Сидоров	4	4	3	3	4	100
УТР02Б	Жуковская	4	3	5	4	4	100
УТР02А	Юматов	3	3	3	3	3	100
УТР02Б	Жукова	5	5	5	5	5	200

Проект решения задачи



Программный модуль «МЕНЮ»:

```
Private Sub CmdMenu1_Click()  
    Load frmVvod  
    frmVvod.Show  
End Sub
```

```
Private Sub CmdMenu2_Click()  
    Load frmVvod  
    frmVvod.Show  
End Sub
```

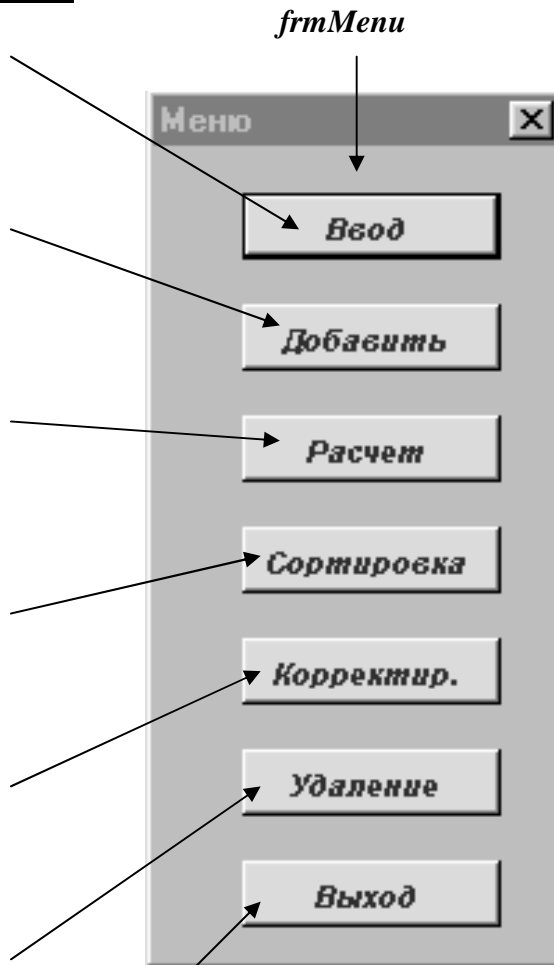
```
Private Sub CmdMenu3_Click()  
    Load frmRas  
    frmRas.Show  
End Sub
```

```
Private Sub CmdMenu4_Click()  
    Load frmSort  
    frmSort.Show  
End Sub
```

```
Private Sub CmdMenu5_Click()  
    Load frmKor  
    frmKor.Show  
End Sub
```

```
Private Sub CmdMenu6_Click()  
    Load frmDel  
    frmDel.Show  
End Sub
```

```
Private Sub CmdMenu7_Click()  
    End  
End Sub
```



Программный модуль «ВВОД» (вызывается и в режиме «ДОБАВИТЬ»):

```
Dim i As Double
```

```
Private Sub UserForm_Activate()
```

```
    ' Подсчет количества строк в таблице и вывод на форму в поле txtN
```

```
    i = 1
```

```
    Do While Cells(i, 1) > " "
```

```
        i = i + 1
```

```
    Loop
```

```
    txtN.Enabled = True
```

```
    txtN.Text = CStr(i - 2)
```

```
    txtN.Enabled = False
```

```
End Sub
```

```
Private Sub CmdVvod_Click()
```

```
    Cells(i, 1) = txtInd.Text
```

```
    Cells(i, 2) = txtFIO.Text
```

```
    Cells(i, 3) = CInt(txtM1.Text)
```

```
    Cells(i, 4) = CInt(txtM2.Text)
```

```
    Cells(i, 5) = CInt(txtM3.Text)
```

```
    Cells(i, 6) = CInt(txtM4.Text)
```

```
    Cells(i, 7) = CInt(txtM5.Text)
```

```
    txtN.Enabled = True
```

```
    txtN.Text = CStr(i - 1)
```

```
    txtN.Enabled = False
```

```
    i = i + 1
```

```
End Sub
```

```
Private Sub CmdCancel_Click()
```

```
    txtInd.Text = "": txtFIO.Text = "": txtM1.Text = ""
```

```
    txtM2.Text = "": txtM3.Text = "": txtM4.Text = ""
```

```
    txtM5.Text = ""
```

```
End Sub
```

```
Private Sub CmdExit_Click()
```

```
    frmVvod.Hide
```

```
End Sub
```

frmVvod

Программный модуль «РАСЧЕТ»:*frmRas***Private Sub CmdRas_Click()***'Ввод обычной стипендии*

St = CSng(txtSt)

i = 2

Do While Cells(i, 1) > " "

' Подсчет двоек по экзаменам

k2 = 0: k4 = 0: k5 = 0

For j = 1 To 5

If Cells(i, j + 2) = 2 Then k2 = k2 + 1

If Cells(i, j + 2) = 4 Then k4 = k4 + 1

If Cells(i, j + 2) = 5 Then k5 = k5 + 1

Next j

'Расчет стипендии

If k5 = 5 Then

Stip = St * 2

Else

If k5 + k4 = 5 Then

Stip = St * 1.3

Else

If k2 > 0 Then

Stip = 0

Else

Stip = St

End If

End If

End If

Cells(i, 8) = Stip

i = i + 1

Loop

End Sub**Private Sub CmdE_Click()**

frmRas.Hide

End Sub

Программный модуль «СОРТИРОВКА»:

Private Sub UserForm_Initialize()

'Заполнение списка ListBox1

```

ListBox1.AddItem "Индекс группы"
ListBox1.AddItem "ФИО"
ListBox1.AddItem "Оценка эк. 1"
ListBox1.AddItem "Оценка эк. 2"
ListBox1.AddItem "Оценка эк. 3"
ListBox1.AddItem "Оценка эк. 4"
ListBox1.AddItem "Оценка эк. 5"
ListBox1.AddItem "Стипендия"

```

End Sub

Private Sub CmdSort_Click()

*'Определение количества
'строк в таблице*

```

N = 1
Do While Cells(N, 1) > " "
    N = N + 1
Loop
N = N - 1
'Номер выбранного критерия
k = ListBox1.ListIndex + 1
'Сортировка

```

i = 2

Do While i <= N

x = Cells(i, k)

kx = i: i = i + 1

Do While i <= N

y = Cells(i, k)

ky = i: i = i + 1

If y < x Then

For j = 1 To 8

r = Cells(kx, j)

Cells(kx, j) = Cells(ky, j)

Cells(ky, j) = r

Next j

x = y

End If

Loop

i = kx + 1

Loop

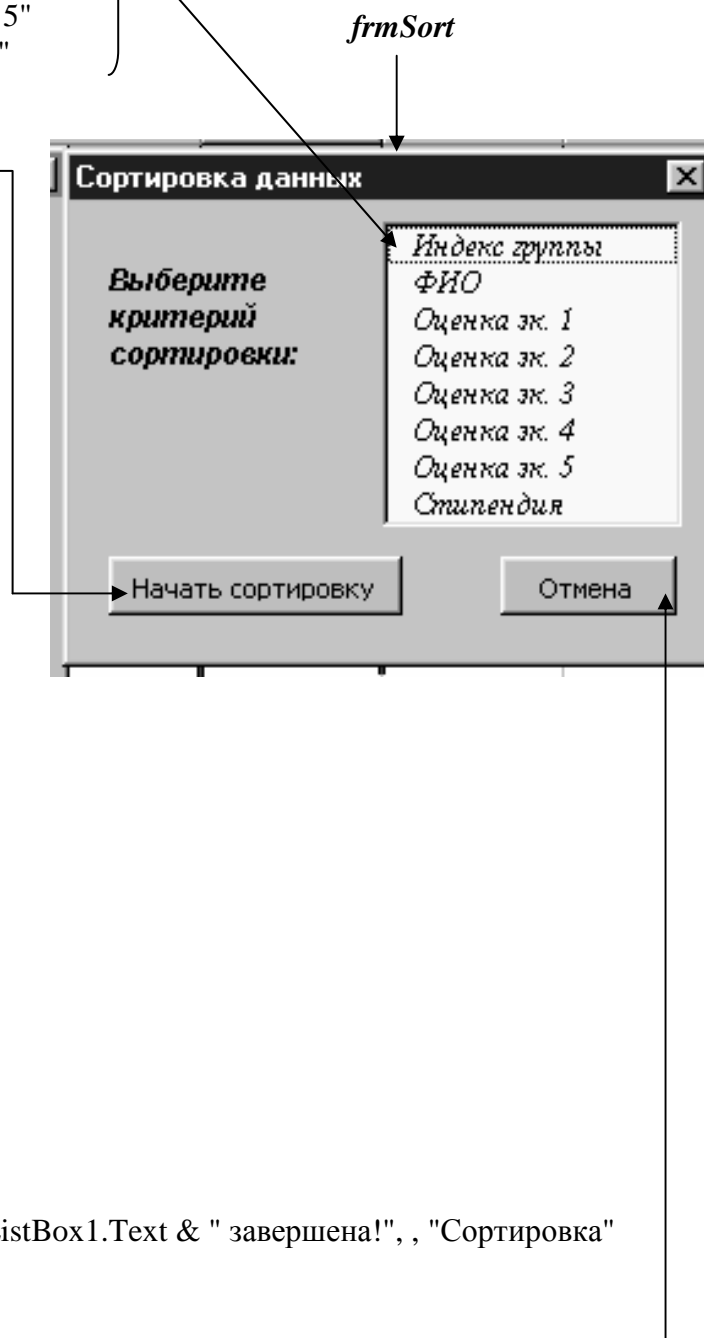
MsgBox "Сортировка по " & ListBox1.Text & " завершена!", , "Сортировка"

End Sub

Private Sub CmdVuh_Click()

frmSort.Hide

End Sub



Программный модуль «КОРРЕКТИРОВКА»:
Модуль KOR

*' Заполнение списка ListBox1
' из колонки таблицы ФИО*

```
Private Sub UserForm_Initialize()
    N = 2
    Do While Cells(N, 2) > " "
        ListBox1.AddItem Cells(N, 2)
        N = N + 1
    Loop
    ListBox1.ListIndex = 0
    ' Очистка текстового поля txtFIO
    txtFIO = ""
End Sub
```

*' Заполнение списка ListBox1
' из колонки таблицы неповторяющимися ФИО*

```
Private Sub UserForm_Initialize()
    N = 2
    Do While Cells(N, 2) > " "
        M = ListBox1.ListCount
        pr = 1
        For j = 1 To M
            ListBox1.ListIndex = j - 1
            If Cells(N, 2) = ListBox1.Text Then pr = 0
        Next j
        If pr = 1 Then ListBox1.AddItem Cells(N, 2)
        N = N + 1
    Loop
    ListBox1.ListIndex = 0
    ' Очистка текстового поля txtFIO
    txtFIO = ""
End Sub
```

Private Sub CmdSeek_Click()

*' Выбор фамилии из списка ListBox1
' или введенной в поле txtFIO*

If Opt1 = True Then Fio = ListBox1.Text Else Fio = txtFIO

' Поиск фамилии в таблице

N = 2: Pr = 0

Do While Cells(N, 2) > " "

 If Cells(N, 2) = Fio Then

 Cells(N, 1).Select

 Pr = 1

 Exit Do

End If

N = N + 1

Loop

If Pr = 1 Then

' Загрузка формы для корректировки

' если фамилия найдена frmKor1

 Load frmKor1

 frmKor1.Show

Else

' Вывод сообщения если фамилия не найдена

 MsgBox "ФИО не найдена", vbCritical, "Корректировка"

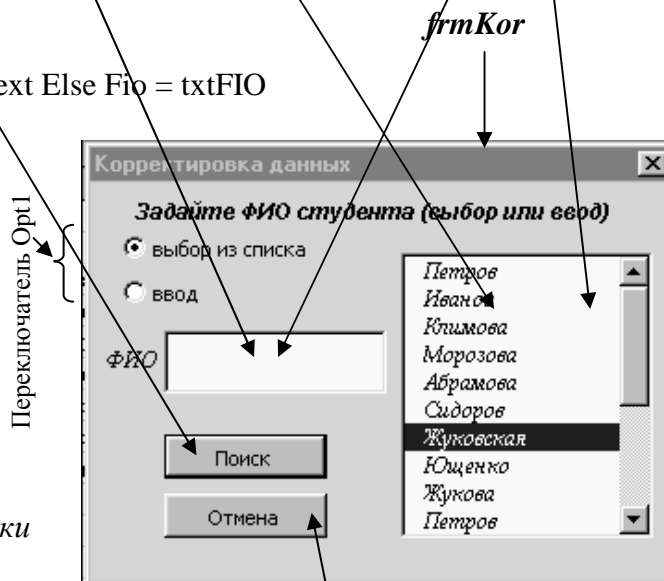
End If

End Sub

Private Sub CmdO_Click()

frmKor.Hide

End Sub



Программный модуль «КОРРЕКТИРОВКА» (продолжение):

Модуль KOR1

' i – номер найденной для корректировки строки в таблице

Dim i As Double

Private Sub UserForm_Activate()

' Вывод данных о студенте

' с найденной фамилией на форму frmKor1

i = ActiveCell.Row

txtInd.Text = Cells(i, 1)

txtFIO.Text = Cells(i, 2)

txtM1.Text = CStr(Cells(i, 3))

txtM2.Text = CStr(Cells(i, 4))

txtM3.Text = CStr(Cells(i, 5))

txtM4.Text = CStr(Cells(i, 6))

txtM5.Text = CStr(Cells(i, 7))

End Sub

Private Sub CmdVvod_Click()

' Ввод новых данных после корректировки

Cells(i, 1) = txtInd.Text

Cells(i, 2) = txtFIO.Text

Cells(i, 3) = CInt(txtM1.Text)

Cells(i, 4) = CInt(txtM2.Text)

Cells(i, 5) = CInt(txtM3.Text)

Cells(i, 6) = CInt(txtM4.Text)

Cells(i, 7) = CInt(txtM5.Text)

End Sub

Private Sub CmdCancel_Click()

' Очистка текстовых полей

txtInd.Text = "": txtFIO.Text = "": txtM1.Text = ""

txtM2.Text = "": txtM3.Text = "": txtM4.Text = ""

txtM5.Text = ""

End Sub

Private Sub CmdExit_Click()

frmKor1.Hide

End Sub

frmKor1

Корректировка данных

Индекс группы: УТР02Б

Фамилия студента: Жуковская

Оценки: Эк1: 4, Эк2: 3, Эк3: 5, Эк4: 4, Эк5: 4

Ввод Очистить Выход

Программный модуль «УДАЛЕНИЕ»:**Private Sub CmdU_Click()**

' N – количество строк в таблице

N = 1

Do While Cells(N, 1) > " "

 N = N + 1

Loop

N = N - 1

' Удаление (u – количество удаленных строк)

i = 2: u = 0

Do While i <= N

 kx = i

' Подсчет количества двоек K2

 k2 = 0

 For j = 1 To 5

 If Cells(i, j + 2) = 2 Then k2 = k2 + 1

 Next j

 i = i + 1

 If k2 >= 2 Then

 u = u + 1

 Do While i <= N

 For j = 1 To 8

 Cells(i - 1, j) = Cells(i, j)

 Next j

 i = i + 1

 Loop

' Удаление последней строки

 Rows(N).Delete

 i = kx: N = N - 1

 End If

Loop

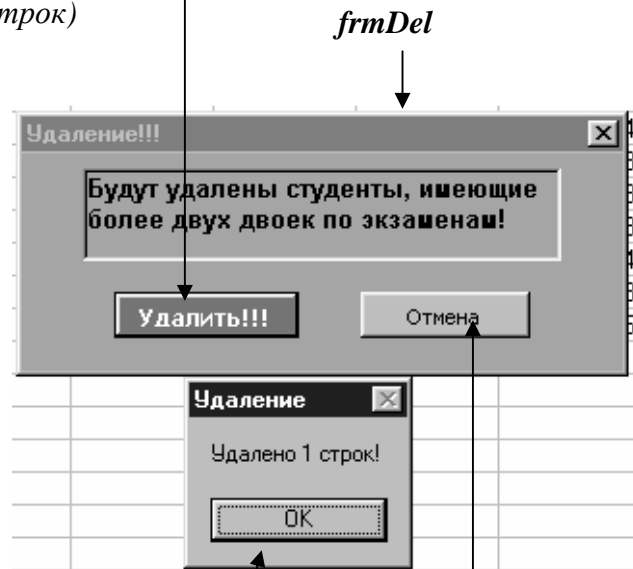
MsgBox "Удалено " & CStr(u) & " строк!", , "Удаление"

End Sub

Private Sub CmdV_Click()

frmDel.Hide

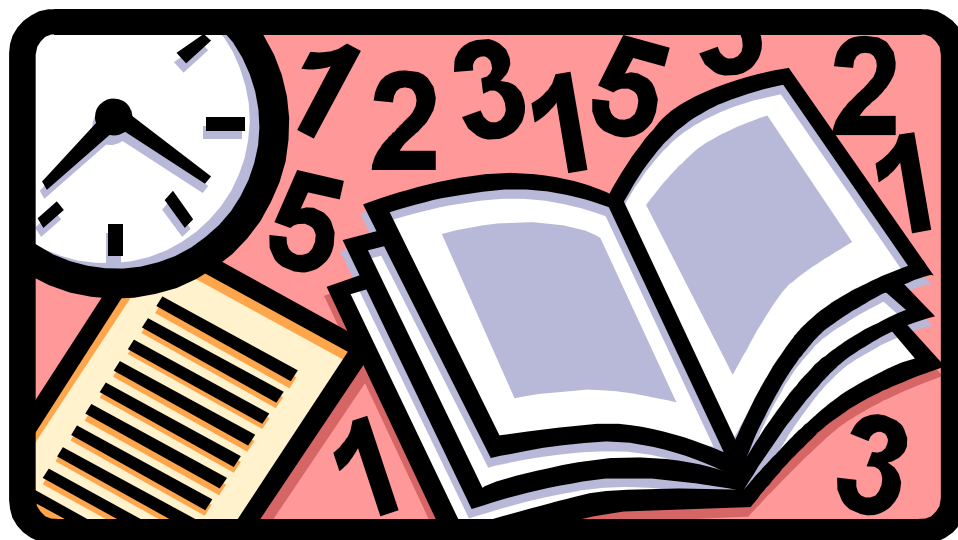
End Sub



ЗАКЛЮЧЕНИЕ

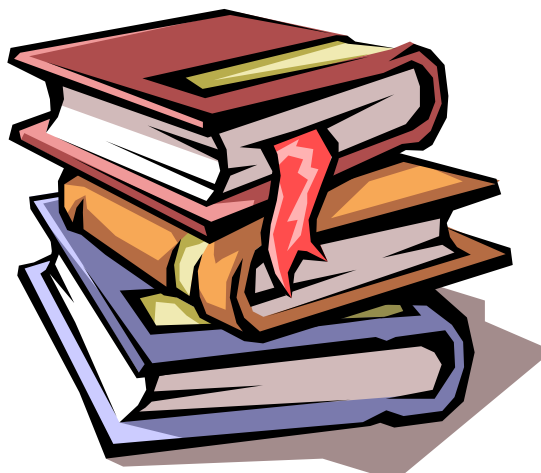
Надеемся, что любознательный пользователь компьютера, разобравшись в приведенном примере, самостоятельно разработав проект, формы пользователя и программные модули задания, используя в качестве справочного материала методические указания данного пособия, путем отладки проекта, добившись желаемых результатов, проникся уважением к возможностям VBA и получил теоретические знания и практические навыки в обработке табличных данных, используемых в различных классах задач.

Авторы будут благодарны читателям за замечания, новые решения и интересные задачи, которые можно направлять как в адрес кафедры "Вычислительной математики и программирования" ФВТИ ДонНТУ, так и по E-mail: shamaev@fvti.dgtu.donetsk.ua.



СПИСОК ЛИТЕРАТУРЫ

1. Браун С. VISUAL BASIC 6: Учебный Курс. – Питер, 1999. – 688с.
2. Visual Basic 6: Полное Руководство. – СПб: ВHV - Санкт - Петербург, 1999. – 992с.
3. Брюс Мак-Кинни Крепкий орешек Visual Basic. // Русская редакция, 1999. – 632с.
4. М. Ченнел Разработка приложений на Microsoft Visual Basic 6.0 // Трейдинг Лимитед, 2000. – 400с.
5. А. Гарнаев Самоучитель VBA. Технология создания пользовательских приложений // ВHV - Санкт - Петербург, 1999. – 512 с.
6. А. Гарнаев Использование MS Excel и VBA в экономике и финансах. – СПб: БХВ – Санкт-Петербург, 2000. – 336 с.
7. Кен Гетц, Майк Джилберт Программирование в Microsoft Office. Руководство по Visual Basic for Applications. – СПб: ВHV, 2000. – 384с.
8. Microsoft Press Руководство программиста по Visual Basic для MS Office 97 // Русская редакция, 1997. – 544с.
9. Deborah Kurata Doing Objects in Microsoft Visual Basic 6 // 1999. – 642 p.



**«РЕШЕНИЕ ЗАДАЧ В MS EXCEL
С ИСПОЛЬЗОВАНИЕМ VBA»
Методические указания и примеры**

(для студентов всех специальностей)

Составители: *Зензеров Владимир Иванович, ст. преподаватель
Славинская Людмила Васильевна, ассистент
Ефименко Константин Николаевич
Добровольский Юрий Николаевич*

Підп. до друку 04.04.2007 р.	Формат 60x84 ¹ / ₁₆ .	Папір офсетний.
Різографічний друк.	Ум.-др. арк. 3,4.	Тираж 100 прим.
	Замовл. № 0404.	

Донецький національний технічний університет
83000 м.Донецьк-00, вул. Артема, 58

Друк з оригінал-макету МПП "ВІК"
Свідоцтво про реєстрацію ДК №382 від 26.03.2001 р.
83059, м. Донецьк, вул. Разенкова, 12/17, тел. (062) 381-70-87



ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«РЕШЕНИЕ ЗАДАЧ В MS EXCEL С ИСПОЛЬЗОВАНИЕМ VBA»

Методические указания и примеры
(для студентов всех специальностей)



