МИНИСТЕРСТВО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский национальный
технический университет

Кафедра «Электроснабжение»

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ VBA

Лабораторный практикум
по дисциплине «Информатика»

Минск
БНТУ
2014

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Кафедра «Электроснабжение»

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ VBA

Лабораторный практикум
по дисциплине «Информатика»
для студентов специальности 1-43 01 03
«Электроснабжение (по отраслям)»

Минск
БНТУ
2014

УДК 681.3.07
ББК 32.973.26-018.2.75
П78

Составитель
А.В. Горностай

Рецензенты:
В.В. Бахтизин, А.В. Манюкевич

Программирование на языке VBA : лабораторный практикум по
П78 дисциплине «Информатика» для студентов специальности 1-43 01 03 «Электроснабжение (по отраслям)» / сост. А.В. Горностай. – Минск : БНТУ, 2014. – 67 с.
ISBN 978-985-550-411-6.

Лабораторный практикум предназначен для студентов 1 курса, начинающих изучать программирование на объектно-ориентированном языке Visual Basic for Application (VBA) практически с нуля. В каждой лабораторной работе студенту предлагается выполнить несколько заданий по определенной теме, пользуясь инструкциями в тексте задания и методическими материалами по теме работы, а также дается задание для дополнительной самостоятельной работы.

ISBN 978-985-550-411-6

Введение

Visual Basic for Applications (VBA) – разновидность объектно-ориентированного языка программирования Visual Basic, встраиваемая практически во все офисные и многие другие приложения. Для использования его возможностей нужно из основного приложения, в котором он встроен, войти в редактор VBA. В приложениях MS Office 2010 (Word, Excel, Acces, Power Point др.) это делается нажатием комбинации клавиш <Alt+F11> или выбором команды *Visual Basic* на ленте вкладки *Разработчик*. В обоих случаях откроется главное окно редактора VBA, в котором и производятся все действия по созданию кода программ и графических объектов приложения VBA.

Для Excel 2010 код программы можно вводить в нескольких местах - на листах VBA в папке *Microsoft Excel Objects*, в модулях в папке *Modules* и в разделах окна кода, связанных с графическим объектом *UserForm*. Для быстрого запуска программы на VBA используется функциональная клавиша <F5>, а для пошаговой проверки программы – клавиша <F8>.

Основная цель лабораторного практикума – научить начинающего пользователя приемам офисного программирования для создания приложений по обработке табличных данных на VBA с одновременным изучением основных синтаксических конструкций и объектов языка, свойств и методов объектов Excel.

В каждой лабораторной работе студенту предлагается выполнить несколько заданий, пользуясь инструкциями и методическими материалами по теме работы, а также задание для самостоятельной работы.

По выполненной лабораторной работе требуется подготовить отчет, который должен содержать:

1. Наименование и цель работы.
2. Отрабатываемые задания и краткое описание хода их выполнения.
3. Пометки наиболее трудных вопросов для дополнительной проработки.
4. Выводы по работе.

Лабораторная работа № 1

Работа с макросами в офисных приложениях

Цель работы: изучить порядок создания, редактирования и выполнения макросов в офисных приложениях.

Задание 1 ***Работа с макросами в редакторе VBA***

1. Создайте новую книгу Excel и сохраните ее с именем *macros.xlsm*.

2. Войдите в редактор VBA, нажав комбинацию клавиш <Alt+F11>.

1. Дважды щелкните левой кнопкой мыши по объекту *Лист1* в окне *Project Explorer* и выберите команду *Procedure* в меню *Insert*.

2. В открывшемся диалоговом окне *Add Procedure* присвойте создаваемому макросу имя *Macros1*, остальные поля окна оставьте без изменений.

3. Закройте окно *Add Procedure* нажатием кнопки *ОК*. В окне кода появятся две строки кода на языке VBA - строка заголовка *Public Sub Macros1()* и строка завершения *End Sub*. Это стандартная заготовка для ввода кода макроса с именем *Macros1* в папку *Microsoft Excel Objects*.

4. Между строками заголовка и завершения макроса введите следующий код тела макроса:

```
Dim a As Integer, b As Integer, s As Integer
a=4
b= 3
s=a + b
MsgBox("Сумма a +b равна " & s)
```

5. Установите курсор в любое место кода макроса и нажмите клавишу <F5>. Макрос выполнится и в стандартном окне на *Лист1* вы увидите результат.

6. Вернитесь в окно кода и запустите макрос, когда курсор не будет находиться внутри кода макроса. Откроется диалоговое окно *Macros*, из которого также можно выполнить записанный макрос.

7. Вставьте стандартный модуль, выбрав команду *Module* в меню *Insert.* В открывшемся окне кода впишите следующий код:

```
Public Sub Macros2()
Dim a As Integer, b As Integer, s As Integer
```

a = InputBox("Введите значение a")
b = InputBox("Введите значение b")
s = a + b
MsgBox ("Сумма a + b равна " & s)
End Sub

8. Протестируйте записанный макрос, запуская его на выполнение командой *Run Sub/UserForm* в меню *Run*.

9. Двойным щелчком по объекту *Эта книга* в окне *Project Explorer* откройте окно кода и вставьте в него код макроса из п. 7, заменив имя макроса на *Macros3*.

10. Протестируйте записанный макрос, запуская его на выполнение щелчком по кнопке  на панели инструментов *Standart* окна редактора VBA.

11.Проанализируйте использованные способы записи и запуска на выполнение макросов в редакторе VBA.

Задание 2 ***Работа с макрорекордером в Excel 2010***

1. Выделите одну или несколько ячеек на *Лист1* окна Excel и включите режим записи макроса.

2. В открывшемся окне *Запись макроса* введите имя макроса *Format1* и назначьте его выполнение по нажатию клавиш <Ctrl+s>.

3. Установите для выделенных ячеек курсивный полужирный шрифт *Arial Narrow* 14-го размера и числовой формат с двумя десятичными цифрами, используя команды вкладки *Главная*.

4. Остановите запись макроса и назначьте его выполнение кнопке на панели быстрого доступа.

5. Протестируйте записанный макрос, выделяя любые ячейки на активном листе книги Excel и запуская его кнопкой с панели быстрого доступа.

6. Запишите макрос с именем *Format2*, возвращающий ячейкам первоначальные параметры форматирования и назначьте его выполнение кнопке на *Лист1* в окне Excel, используя команду *Вставить* на вкладке *Разработчик*.

7. Протестируйте поочередно макросы *Format1* и *Format2*.

8. Сохраните результаты работы и оформите отчет по лабораторной работе.

Задание для самостоятельной работы

Запишите макрос в приложении Word 2010, при выполнении которого применяется курсивный полужирный шрифт ***Times New Roman*** 14 размера к выделенным фрагментам текста активного документа. Назначьте его выполнение комбинации клавиш и кнопке на панели быстрого доступа.

Методические материалы

Макрос - это запрограммированная последовательность действий, записанная на языке программирования VBA. Макрос можно запускать сколько угодно раз, заставляя Excel выполнять последовательность любых нужных нам действий.

Для создания макроса в редакторе VBA необходимо сначала открыть специальное окно - *Microsoft Visual Basic for Applications* из окна офисного приложения, в котором мы хотим создавать макрос. Это окно называется редактор программ на VBA , или кратко - редактор VBA. Во всех офисных приложениях это можно сделать одинаково – нажать <Alt+F11>. Откроется главное окно редактора VBA, возможный вид которого показан на рис.1.1.

Макросы хранятся в программных модулях. В любой книге Excel можно создать любое количество программных модулей и разместить там макросы. Один модуль может содержать любое количество макросов. Выбор макроса в модуле для запуска на выполнение производится установкой курсора в любое место текста кода макроса. Доступ ко всем модулям осуществляется с помощью окна *Project Explorer* в левом верхнем углу окна редактора VBA (если его не видно, нажмите <Ctrl+R>).

Программные модули бывают нескольких типов для разных ситуаций:

- стандартные модули - используются в большинстве случаев, когда речь идет о макросах. Для создания такого модуля выберите в меню I*nsert* команду *Module*. В появившееся окно кода нового пустого модуля можно вводить команды на VBA, набирая их с клавиатуры или копируя их из другого модуля;

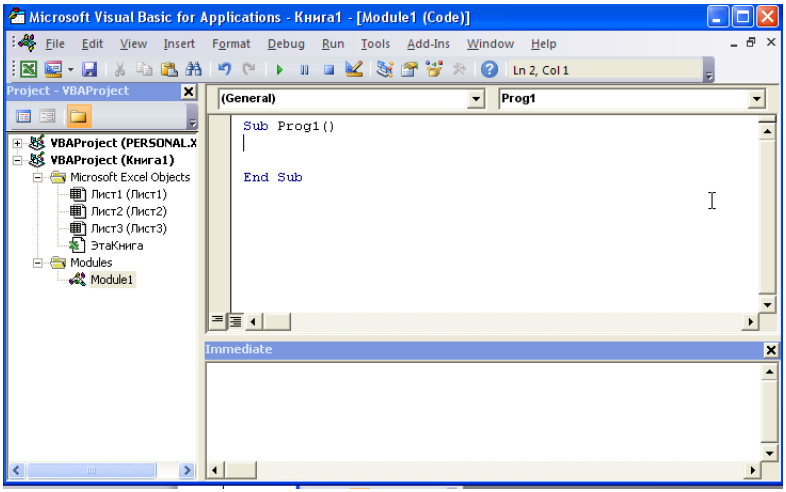


Рис. 1.1 Вид окна редактора VBA в Excel 2010

- модуль *Эта книга* также виден в окне *Project Explorer*. В этот модуль обычно записываются макросы, которые выполняются при наступлении каких-либо событий в книге (открытие или сохранение книги, печать файла и т.п.);
- модуль листа - доступен через *Project Explorer* и через контекстное меню листа Excel (команда *Исходный текст*). Сюда записывают макросы, которые должны выполняться при наступлении определенных событий на листе.

Вид макроса с именем *Macros_1*, введенного в модуль листа *Лист1*, приведен на рис. 1.2.

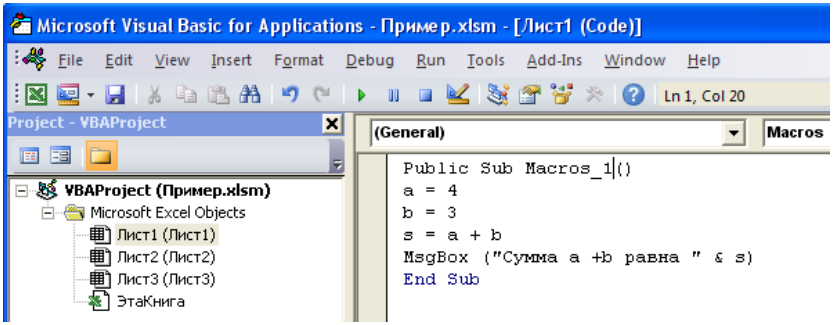


Рис. 1.2 Пример кода макроса в модуле *Лист1*

Структура кода этого макроса следующая:

- любой макрос начинается с оператора *Sub*, за которым идет имя макроса и список аргументов в скобках. Если аргументов нет, то скобки надо оставить пустыми;
- любой макрос должен заканчиваться оператором *End Sub;*
- все, что находится между *Sub* и *End Sub* – это команды, которые будут выполняться при запуске макроса.

Такой способ создания макросов подходит для пользователей, имеющих опыт программирования на VBA. Для начинающих программистов существует другой способ создания макросов – использование макрорекордера.

Макрорекордер - это небольшая программа, встроенная в Excel, которая переводит любое действие пользователя на язык программирования VBA и записывает код в программный модуль.

Естественно, у такого способа есть свои плюсы и минусы:

- макрорекордер записывает только те действия, которые выполняются в пределах окна Excel;
- макрорекордер может записать только те действия, для которых есть команды меню или кнопки в Excel. Программист же может написать макрос, который делает то, что Excel никогда не сделает;
- допущенная во время записи макроса ошибка будет записана. Однако можно нажать на кнопку отмены последнего действия *Undo* - во время записи макроса макрорекордером она не просто возвращает в предыдущее состояние, но и стирает последнюю записанную команду на VBA.

Запуск макрорекордера в Excel 2010 осуществляется кнопкой *Запись макроса*, которую можно найти в нескольких местах:

- в левой части строки состояния книги Excel;
- на вкладке *Вид* при выборе команды *Макросы;*
- на специальной вкладке *Разработчик*, где собраны все инструменты для работы с макросами.

При нажатии на кнопку *Запись макроса* открывается диалоговое окно *Запись макроса*, показанное на рис. 1.3.

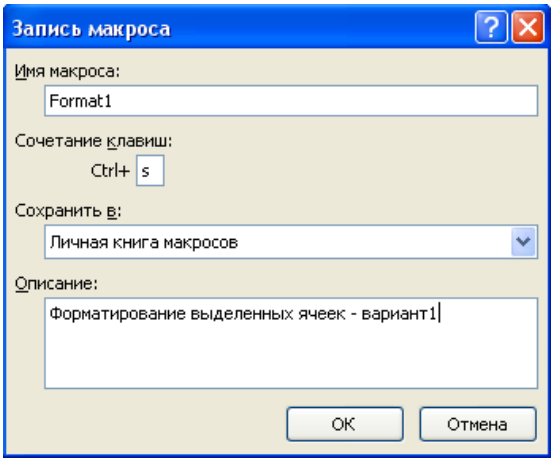


Рис. 1.3 Диалоговое окно *Запись макроса*

В поле *Имя макроса* необходимо ввести имя макроса, которое должно начинаться с буквы.

В поле *Сочетание клавиш* можно ввести букву для быстрого вызова макроса.

Выбором значения поля *Сохранить в* определяется, где будет сохранен макрос:

- *Эта книга* - макрос сохраняется в модуль текущей книги и будет выполняться , пока эта книга открыта в Excel;
- *Новая книга* – макрос сохраняется в шаблон, на основе которого создается любая новая пустая книга в Excel;
- *Личная книга макросов* – это специальная книга Excel с именем *Personal.xlsb*, которая используется как хранилище макросов. Все макросы из *Personal.xlsb* загружаются в память при старте Excel и могут быть запущены в любой момент и в любой книге.

В поле *Описание* желательно внести краткое описание макроса, чтобы в будущем легко было вспомнить его назначение.

После нажатия кнопки *ОК* начинается запись выполняемых действий пользователя. Во время записи макроса кнопка *Запись макроса*  изменяет свое назначение на *Остановить запись*  , которую и необходимо нажать для остановки записи макроса.

Для просмотра и редактирования кода записанного макроса используется кнопка *Изменить* окна *Макрос,* показанная на рис. 1.4.

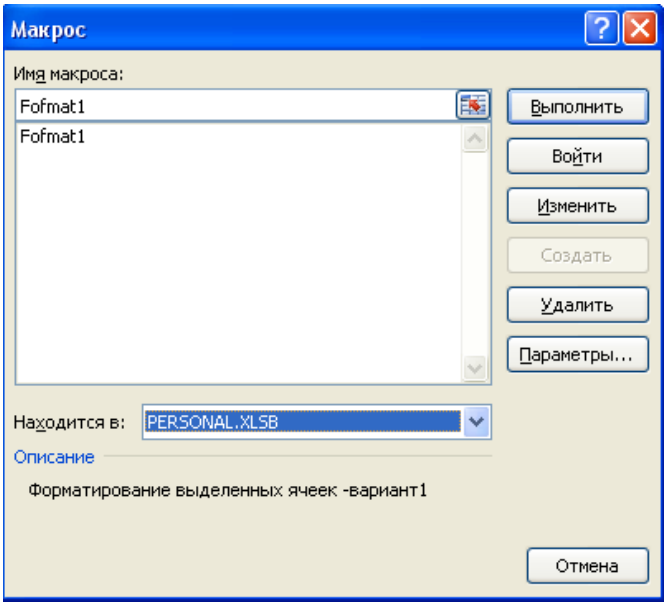


Рис. 1.4. Окно работы с макросами

Окно можно открыть нажатием кнопки *Макросы* на вкладках *Разработчик* или *Вид* или комбинацией клавиш <Alt+F8>.

Из окна *Макрос* можно выполнить и другие действия с записанными макросами, нажимая на требуемую кнопку (например, удалить ненужный макрос). При нажатии кнопки *Параметры* откроется диалоговое окно, в котором можно переопределить клавишу запуска на выполнение и отредактировать описание макроса.

Для запуска макроса непосредственно из окна приложения Excel нужно вставить кнопку с панели *Элементы управления формы* и настроить ее на запуск требуемого макроса.

Для настройки макроса кнопке на панели быстрого доступа в Excel 2010 нужно выполнить следующие операции:

- кликнуть мышкой на направленную вниз стрелку на панели быстрого доступа, в открывшемся меню выбрать пункт *Другие команды*;
- в открывшемся окне *Параметры Excel* в списке *Выбрать команды из* выбрать *Макросы*;

- в поле, расположенном ниже, выбрать имя макроса и нажать кнопку *Добавить*. Имя выбранного макроса появится в списке на правом поле окна;
- в списке настройки этого поля выбрать имя макроса и нажать кнопку *Изменить*;
- в открывшемся окне выбрать понравившийся значок и нажать кнопку *ОК*. Выбранный значок появится на панели быстрого доступа;
- закрыть окно *Параметры Excel*.

Настроить макрос кнопке на панели быстрого доступа можно также из меню *Файл*, выбрав команду *Параметры,* и в открывшемся окне *Параметры Excel* выбрать команду *Строка быстрого доступа*.

Настройка макроса кнопке быстрого доступа в Word 2010 производится также, как и в Excel 2010.

Лабораторная работа № 2

Создание и выполнение программ в редакторе VBA

Цель работы: ознакомиться с назначением и интерфейсом окон редактора VBA. Изучить приемы записи, выполнения и отладки кода программы на VBA.

Задание 1 ***Работа с окнами редактора VBA***

1. Создайте новую книгу в приложении Excel и сохраните ее с именем *Program_VBA.xlsm*.
2. Войдите в редактор VBA, научитесь управлять видимостью окон редактора с помощью команд меню *Viev*.
3. Прочитайте методические указания и ознакомьтесь с названиями и назначением всех окон редактора VBA.
4. Изучите информацию о главных окнах редактора по работе с VBA-программами - *Project Explorer* (проводник проектов), *Code* (окно программного кода), *UserForm* (окно формы), *Toolbox* (окно элементов управления), *Properties* (окно свойств).

Задание 2 ***Создание, выполнение и отладка программ на VBA***

1. Щелкните правой кнопкой мыши по имени проекта *VBAProject (Книга1)*. Из открывшегося контекстного меню выберите команду *VBAProject Properties* и присвойте проекту новое имя *VBA_Prog.*
2. Вставьте новый модуль в проект *VBA_Prog*, выполнив команду *Module* в меню *Insert*.
3. Выполните команду *Procedure* в меню *Insert* и присвойте ей имя *Prog1* в открывшемся окне *Add Procedure*. Значения остальных полей окна оставьте без изменения.
4. В окне кода между строчками *Private Sub Prog1()* и *End Sub* введите следующий код:

```
Dim a As Integer, b As Integer, s As Integer
a = 5
b = 8
Cells(3,3)=a
Cells(3,4)=b
s= Cells(3,3)+ Cells(3,4)
Cells(3,5)=s
```

MsgBox ("Сумма значений ячеек C3 и C4 = " & Cells(3,5))

5. Запустите программу *Prog1* на выполнение (<F5>). Программа выполниться и в окне книги Excel вы увидите результат.

6. Вернитесь в окно редактора VBA и запустите программу на выполнение в режиме отладки, нажав клавишу <F8>. После каждого нажатия клавиши <F8> будут последовательно выполняться операторы кода программы.

7. Изучите способы запуска на выполнения и отладки программы VBA с использованием соответственно команд меню *Run* и *Debug*, кнопок панели инструментов *Standart*.

Задание 3 ***Использование диалогового окна InputBox***

1. По аналогии с п. п. 3-4 задания 2 вставьте новую процедуру с именем *Prog2* и введите между строками *Private Sub Prog2()* и *End Sub* следующий код:

```
Dim a As Integer, b As Integer, s As Integer
a = InputBox("Введите значение  a")
Range("D4") = InputBox("Введите значение  b")
Range("C4") = a
s = Cells(4, 3) + Cells(4, 4)
Cells(4, 5) = s
MsgBox ("Значения a и b введены в ячейки C4 и D4")
MsgBox ("Сумма a + b введена в ячейку E4 и равна " & s)
```

2. Выполните действия по пунктам 5 - 6 задания 2 для созданной программы *Prog2*, оцените различия в функциональности программ *Prog1* и *Prog2*.

3. Обратите внимание на разные способы записи и чтения данных в ячейках листа Excel, реализованные на языке VBA (строки 3-6 кода программы п.1).

4. Сохраните результаты работы и оформите отчет по лабораторной работе.

Задание для самостоятельной работы

Реализуйте запуск на выполнение программы VBA по нажатию командной кнопки, вставляемой на рабочем листе Excel с использованием инструментов вкладки *Разработчик*.

Методические материалы

Во многих ситуациях макрорекордер может быть полезным, но в реальной работе только им обойтись невозможно. Полные возможности программирования в офисных приложениях раскрываются при использовании редактора VBA и при решении сложных задач без него не обойтись.

Вход в редактор VBA во всех приложениях MS Office выполняется одинаково:

- самый простой способ: на вкладке *Разработчик* нажать кнопку *Visual Basic*;
- самый быстрый способ: нажать <Alt>+<F11>;
- можно вызвать редактор при возникновении ошибки в макросе;
- можно открыть готовый макрос на редактирование в диалоговом окне *Макрос*.

В любом случае откроется окно редактора VBA, показанное на рис.1.1.

Всего в редакторе VBA 9 окон:

- *Project Explorer* - окно проводника проекта. По умолчанию оно открыто и находится в левой части окна редактора VBA. В нем можно просмотреть компоненты проекта и выполнить с ними множество операций;
- *UserForm* - окно формы. Появляется тогда, когда вы редактируете пользовательскую форму из окна дизайнера формы;
- *Toolbox* - панель инструментов управления. Из нее можно добавить требуемые элементы управления на форму или в документ. Появляется в окне дизайнера форм при создании новой формы;
- *Properties* - одно из самых важных окон. Через него можно просмотреть свойства выбранного элемента управления или компонента проекта и отредактировать их при необходимости;
- *Code* - окно программного кода. В этом окне выполняется основная работа по написанию кода программы на VBA. При открытии программного модуля открывается автоматически;
- *Object Browser* - обозреватель объектов. Необходим для получения информации о классах и объектах, доступных программе;

- *Watches* - окно контролируемых выражений. Используется во время отладки для отслеживания значений выбранных переменных программы и выражений;
- *Locals* - окно локальных переменных. Нужно для отслеживания во время отладки значений переменных текущей процедуры;
- *Immediate* - окно просмотра значений переменных с возможностью выполнения отдельных строк программного кода при отладке и немедленного вывода результата.

Найти любое окно, если его не видно на экране, можно очень просто: нужно выбрать его в меню *View*, и если оно было скрыто, оно появится на экране.

Окно *Project Explorer* (Проводник проектов) при первой активации редактора VBA обычно открыто. Если оно случайно было закрыто, то вызвать его можно:

- нажав на клавиши <Ctrl>+<R>;
- нажав на кнопку *Project Explorer* на панели инструментов *Standard*;
- воспользовавшись командой *Project Explorer* меню *View*.

В окне *Project Explorer* представлено дерево компонентов приложения VBA. Самый верхний уровень – это *Project*, которому соответствует документ Word, рабочая книга Excel, презентация PowerPoint и прочие файлы, с которыми работает данное приложение. Если редактор VBA открыт из Excel, то в *Project Explorer* будут открытые книги Excel и специальная скрытая книга *PERSONAL.XLSB*.

Каждый проект - это одновременно контейнер для хранения как самого документа, так и стандартных модулей, модулей классов и пользовательских форм. Добавить в проект каждый из этих компонентов можно при помощи меню *Insert* или через контекстное меню в *Project Explorer*.

Стандартный модуль - это просто блок с текстовым представлением команд VBA. В нем может быть только два раздела:

- раздел объявлений уровня модуля (объявление переменных и констант уровня модуля);
- раздел методов модуля (расположение процедур и функций).

В большинстве проектов VBA используется только один стандартный модуль, куда и записывается весь код программы. Созда-

вать новые стандартные модули есть смысл только из следующих соображений:

- для удобства экспорта и импорта (из контекстного меню в *Project Explorer*). Так можно очень удобно обмениваться блоками кода между приложениями VBA;
- для повышения производительности. При вызове любой процедуры модуля происходит компиляция всего модуля, поэтому иногда выгоднее разместить процедуры в разных модулях, чтобы компилировать только нужный в данный момент код;
- для улучшения читаемости кода. Если приложение выполняет разные группы задач, то код, относящийся к каждой группе, лучше поместить в свой модуль.

Пользовательские формы являются одновременно хранилищем элементов управления и программного кода, который относится к ним, самой форме и происходящими с ними событиями.

В окне *VBAProject Properties, которое* в контекстном меню *VBAProjec,* можно:

- изменить имя проекта;
- ввести описание проекта, информацию о файле справки и параметры, которые будут использоваться компилятором;
- защитить проект паролем.

Если необходимо создать макрос вручную, а макросов в данном проекте еще нет, то нужно щелкнуть правой кнопкой мыши по имени проекта и в контекстном меню выполнить команду *Insert>Module*. В проекте будет создан новый модуль и сразу открыт в окне редактора кода.

Если макросы уже созданы в этом проекте (макрорекордером или вручную), то модуль будет уже создан. Его можно увидеть под контейнером *Modules*.

В окне *Code* (Редактор программного кода) выполняется основная часть работы по программированию, поэтому знать приемы работы с ним нужно очень хорошо. Открыть окно редактора кода можно разными способами:

- выбрать нужный элемент (в *Project Explorer*, в дизайнере форм и т.п.) и в контекстном меню выбрать *View>Code*;
- нажать клавишу <F7>;
- выбрать команду *Code* из меню *View*;

- дважды щелкнуть по объекту модуля в *Project Explorer* (или выделить его и нажать на кнопку *Enter*).

В верхней части окна редактора кода находятся два списка. Список слева – это список объектов. В нем можно выбрать объект, к которому будет относиться код. Если открыт программный код модуля, то здесь будет только пункт *General*. Если открыта форма - в этом списке кроме *General* можно выбрать саму форму или любой ее элемент управления и записать (просмотреть, изменить, удалить) для него код.

Список справа – это список процедур или событий. В нем есть раздел *Declarations* - объявления уровня всего модуля и список всех процедур (макросов) для стандартного модуля или событий, если создается или открыт код для формы. При выборе нужного события автоматически создаются строка заголовка и строка окончания процедуры для обработки этого события. Между этими строками и нужно вставить исполняемый при возникновении данного события код.

Лабораторная работа № 3

Проектирование форм в редакторе VBA

Цель работы: научиться работать с формами и элементами управления для создания графического интерфейса программ VBA, изучить основные свойства и методы формы и основных элементов управления, размещаемых на форме.

Задание 1 ***Проектирование форм в VBA-программах***

1. Откройте приложение Excel и создайте новую книгу. Сохраните ее с именем *VBA_Form.xlsm*.
2. Войдите в редактор VBA. Вставьте новый модуль и, используя команду *UserForm* меню *Insert* (или контекстное меню модуля), вставьте в окно кода редактора новую форму. По умолчанию ей будет присвоено имя *UserForm1*. Рядом откроется окно элементов управления *ToolBox*.
3. Разместите на форме элементы управления *Label*, *TextBox* и *CommandButton* с панели *ToolBox*, как показано на рис. 3.1.
4. Используя окно *Properties*, отредактируйте свойства формы и элементов управления, как показано на рис. 3.2.

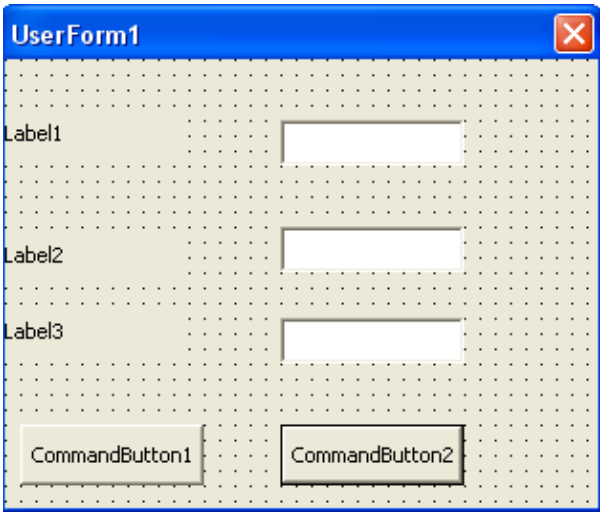


Рис. 3.1 Вид формы после вставки элементов управления

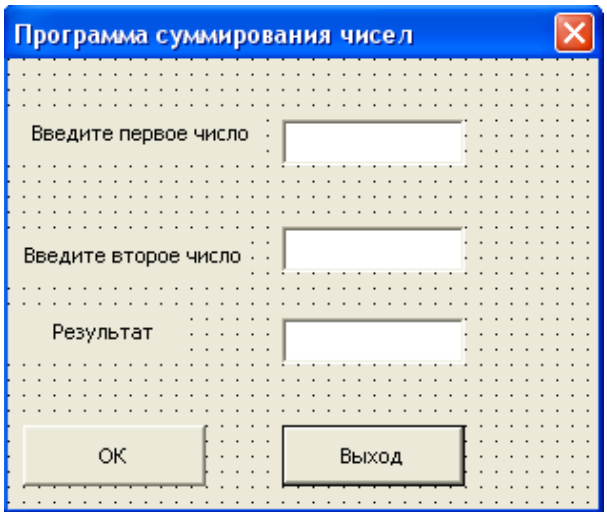


Рис. 3.2 Вид формы после редактирования

Задание 2 ***Программирование форм в VBA***

1. Двойным щелчком по кнопке *OK* на форме откройте окно кода и меду строчками *Private Sub CommandButton1_Click()* и *End Sub* введите код обработчика события - нажатия пользователем кнопки *OK* на форме при выполнении программы:

Dim a As Integer, b As Integer, s As Integer
a=Val(TextBox1)
b=Val(TextBox2)
s = a + b
TextBox3=Str(s)

2. По аналогии с п. 1 введите код обработчика события - одиночный клик мышкой по кнопке *Выход* при выполнении программы:

UserForm1.Hide

3. Запустите программу на выполнение (<F5>) из окна редактора кода или из окна дизайнера форм. В окне основного приложения (книга *VBA_Form*) откроется диалоговое окно формы.

4. Введите исходные данные в поля *TextBox1* и *TextBox2* и нажмите кнопку *OK*. В поле *TextBox3* при отсутствии ошибок в программе появится результат суммирования введенных чисел.

5. Для завершения программы нажмите кнопку *Выход*.

6. Добавьте на форму третью кнопку с названием *Очистка*. Действуя по аналогии с п. 1, введите между строчками *Private Sub CommandButton3_Click()* и *End Sub* следующий код для очистки значений текстовых полей:

TextBox1=" "

TextBox2=" "

TextBox3=" "

7. Протестируйте созданную VBA-программу и сравните ее с программой *Prog2* из предыдущей лабораторной работы, в которой использовались встроенные диалоговые окна *InputBox* и *MsgBox*.

8. Создайте папку на диске *D* с именем *Фамилия_Группа*, выберите команду *Export File* в меню *File* и сохраните форму с именем *Form1.frm* в эту папку. Вместе с формой будут сохранены процедуры для всех размещенных на ней объектов. В дальнейшем все результаты своей работы сохраняйте в свою личную папку.

9. Закройте книгу *VBA_Form.xlsm* с сохранением изменений.

10.Создайте новую книгу Excel и откройте окно редактора VBA.

11.Импортируйте сохраненную форму *Form1.frm* командой *File>Import File* и протестируйте ее.

12. Оформите отчет по лабораторной работе.

Задание для самостоятельной работы

Спроектируйте несколько вариантов оформления внешнего вида созданной формы, используя свойства формы и элементов управления *Label*, *TextBox*, *CommandButton.*

Методические материалы

Для создания графического интерфейса VBA-программ в виде пользовательских диалоговых окон используются формы (объект *UserForm*).

Как правило, форма запускается при открытии пользователем документа, в котором она встроена, или нажатием кнопки в окне этого документа. Пользователь выполняет на ней какие-то действия по вводу или выбору информации, а потом нажимает на кнопку на форме. Программа обрабатывает нажатие на кнопку (событие) с помощью специальной процедуры (обработчик события), записанной при проектировании формы.

Создать форму просто: для этого достаточно в редакторе VBA выполнить команду *Insert>User Form* или щелкнуть правой кнопкой мыши по проекту в окне *Project Explorer* и в контекстном меню выбрать *Insert>UserForm*. Откроются два новых окна: окно дизайнера форм, в котором будет представлено пустое поле формы, и окно *Toolbox* - панель с набором стандартных элементов управления. Возможный вариант окна дизайнера форм представлен на рис. 3.3.

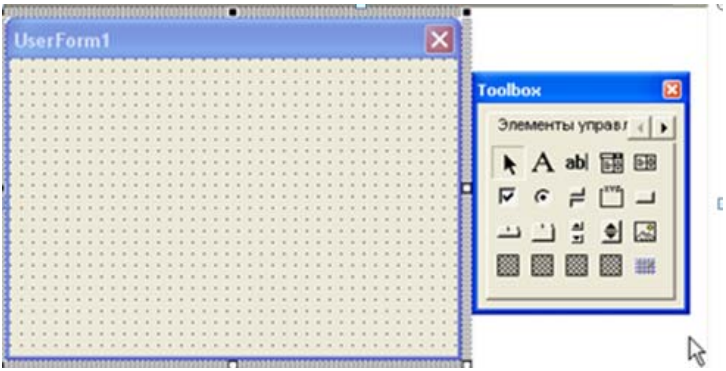


Рис. 3.3 Окно дизайнера форм VBA в Excel

Если окно *Toolbox* по какой-либо причине скрыто, его можно открыть из меню *View*.

Если включен показ окна свойств *Properties* (он включается по клавише <F4>), то в этом окне будут представлены свойства формы. Переход на код для этой формы (по умолчанию открывается событие *Click*) - по клавише <F7>, возврат обратно в окно дизайнера форм - по <Shift>+<F7>.

Некоторые важные свойства объекта *UserForm*:

- свойство *Name* - определяет имя формы, которое используется в программном коде для этой формы;
- свойство *Caption* - определяет заголовок формы;
- свойство *Enabled* - используется для временного отключения формы, если установить *False*;
- свойство *ShowModal* - по умолчанию установлено *True*. Пользователь не может перейти к другим формам или вернуться в документ, пока не закроет эту форму.

Некоторые важные методы объекта *UserForm*:

- *Show* - если форма уже была загружена в память, она просто станет видимой, если еще нет - то будет автоматически загружена;
- *Hide* - форма будет убрана с экрана, но останется в памяти.

Важнейшая концепция языка VBA - события. *Событие* - это что-то, что происходит с программой и может быть ей распознано. Например, к событиям относятся щелчки мышью, нажатия на клавиши, открытие и закрытие форм, перемещение формы по экрану и т.п. Язык VBA построен таким образом, чтобы создавать на нем программы, управляемые событиями.

Самые важные события объекта *UserForm*:

- *Initialize* - происходит при подготовке формы к открытию (появлению перед пользователем);
- *Click* (выбирается по умолчанию) - реакция на одиночный щелчок мыши;
- *Error* - это реакция на возникновение ошибки в форме, используется как возможность предоставить пользователю исправить сделанную им ошибку;
- *Terminate* - событие используется при нормальном завершении работы формы и выгрузке ее из памяти.

Поскольку форма - это во многом просто контейнер для хранения других элементов управления, главное ее событие - *Initialize*. Все остальные события обычно используются не для формы, а для расположенных на ней элементов управления.

Элементы управления - это специализированные объекты, которые можно размещать на формах VBA и непосредственно в документах, используемые для организации взаимодействия с пользователем. В VBA можно использовать как стандартные элементы управления (*CommandButton*, *CheckBox*, *OptionButton* и др.), так и нестандартные - любые другие, которые есть на компьютере (например, *Calendar*). Элементы управления реагируют на события, которые генерирует пользователь (нажатие на кнопку, ввод значения, перемещение ползунка и др.).

Добавление элементов управления на форму производится в дизайнере форм при помощи *Toolbox*. Для этого необходимо выбрать элемент управления в *Toolbox* и перетащить его на форму.

Чаще всего элемент управления *Label* (Надпись) используется как строка состояния с объяснением того, что сейчас произошло или что должен сделать пользователь.

Самое важное свойство элемента *Label* - это *Caption*. В его поле набирается текст, который будет выводиться на форме. Большая часть остальных свойств относится к форматированию этого текста или настройке внешнего вида этого элемента на форме.

Элемент управления *TextBox* (Текстовое поле) - один из самых часто используемых элементов управления. Текстовое поле используется:

- для приема каких-либо текстовых данных, вводимых пользователем;
- для вывода пользователю текстовых данных с возможностью их редактирования или только копирования и печати.

Некоторые важные свойства *TextBox*:

- *Value* или *Text* - то текстовое значение, которое содержится в этом поле. Используется для занесения исходного значения и для приема значения, введенного пользователем, в строковую переменную;
- *AutoSize* - возможность для текстового поля автоматически менять свой размер, чтобы вместить весь текст;
- *Enabled* –установка в *False* блокирует возможность редактирования текста в поле пользователем;
- *Locked* - поле будет выглядеть как обычно, пользователь сможет выделять и копировать данные из него, но не изменять их;
- *MultiLine* – определяет, можно ли использовать в текстовом поле несколько строк;
- *PasswordChar* – позволяет указать, за каким символом будут "прятаться" вводимые пользователем значения. Используется при вводе пароля;
- *ScrollBars* – определяет, будут ли показаны горизонтальная и вертикальная полосы прокрутки (в любом сочетании).

Главное событие для *TextBox* - это *Change* (то есть изменение содержания поля). Обычно на это событие привязывается проверка вводимых пользователем значений или синхронизация введенного значения с другими элементами управления.

Элемент управления *ComboBox* (Поле со списком) обычно используется в двух ситуациях:

- когда пользователю необходимо выбрать одно или несколько значений из списка размером от 4-х до нескольких десятков позиций;
- когда список позиций для выбора необходимо формировать динамически на основании данных из источника (базы данных, листа Excel).

Для заполнения списка позициями используется специальный метод *AddItem*. Обычно он помещается в обработчик события *Initialize* для формы. Применение его может выглядеть так:

```
Private Sub UserForm_Initialize()
ComboBox1.AddItem "Минск"
ComboBox1.AddItem "Могилев"
ComboBox1.AddItem "Витебск"
ComboBox1.AddItem "Гродно"
End Sub
```

Самым важным свойством *ComboBox* является *Value* (или *Text*) - позволяет программным способом установить выбранное значение в списке или вернуть выбранное или введенное пользователем значение.

Главное событие для *ComboBox* - *Change*. Обычно в обработчике этого события проверяются введенные пользователем значения, затем эти значения переносятся в текстовое поле.

Элемент управления *ListBox* (Список) очень похож на *ComboBox*, но применяется гораздо реже по двум причинам:

- в нем нельзя открыть список значений по ниспадающей кнопке, все значения видны сразу в поле и поэтому большое количество позиций в нем уместить трудно;
- пользователь не может вводить свои значения – можно только выбирать из готового списка.

Основные свойства, методы и события у *ListBox* - те же, что и у *ComboBox*. Главное отличие - то, что имеется свойство *MultiSelect*, которое позволяет пользователю выбирать несколько значений. По умолчанию это свойство отключено.

Основные свойства элемента управления *CheckBox* (Флажок):

- *Caption* - надпись справа от флажка, которая объясняет, что выбирается этим флажком;

- *TripleStat*e - если в *False* (по умолчанию), то флажок может принимать только два состояния: установлен или нет. Если для *TripleState* установить значение *True*, то появляется третье значение: *Null*;
- *Value* - само состояние флажка. Может принимать значения *True* (установлен), *False* (снят) и *Null* (нейтральный), когда свойство *TripleState* установлено в *True.*

Главное событие *CheckBox* - *Change*.

Элемент управления *ToggleButton* (Выключатель) выглядит как кнопка, которая при нажатии становится включенной, а при повторном нажатии отключается. У нее могут быть те же два (или три, в соответствии со свойством *TripleState*) состояния, что и у *CheckBox*. Свойства и методы - те же самые. Единственное отличие - в восприятии их пользователем. Обычно *ToggleButton* воспринимается пользователем как переход в какой-то режим или начало выполнения продолжительного действия.

Если *CheckBox* предназначен для выбора не взаимоисключающих вариантов, то *OptionButton* (Переключатель) как раз нужен для выбора варианта в ситуации или/или.

Главных свойств у этого элемента управления два:

- *Caption* - надпись для переключателя;
- *Value* - установлен флажок или нет (только два состояния *True* или *False)*.

Frame (Рамка) - это просто рамка, которая выделяет прямоугольную область на форме и позволяет сгруппировать элементы управления. Помещенные внутрь рамки переключатели считаются взаимоисключающими. При желании рамку можно сделать невидимой, установив для свойства *BorderStyle* значение 1 и убрав значение свойства *Caption*.

Элемент управления *CommandButton* (Кнопка) — самый распространенный элемент управления в формах. В большинстве форм обязательно будет по крайней мере две кнопки: *Cancel* (Отмена) и *OK*. По нажатию кнопки *Cancel* форма должна закрыться, по нажатию кнопки *OK* должно выполниться то действие, ради чего создавалась эта форма.

Главное событие для кнопки - это *Click*. Как правило, к этому событию и привязывается тот программный код (обработчик), ради которого создавалась кнопка.

Самые важные свойства кнопки:

- *Cancel* - если для него установить значение *True*, то это значит, что кнопка будет нажиматься при нажатии на клавишу <Esc>;
- *Caption* - надпись, которая будет на кнопке;
- *Default* - такая кнопка будет считаться нажатой, если пользователь нажал на клавишу <Enter>, а курсор находился в другом месте формы (но не на другой кнопке). Обычно такие кнопки являются главными, по которым выполняется действие, ради которого создавалась форма;
- *Picture* - если просто надпись вас не устраивает, можно назначить кнопке рисунок.

ScrollBar (Полоса прокрутки) чаще всего встречается в текстовых полях, когда введенный текст полностью на экране не умещается.

Главное событие для этого элемента управления - *Change*.

Главные свойства выглядят так:

- *Max* и *Min* - максимальное и минимальное значения, которые можно задать при помощи этого элемента управления;
- *LargeChange* и *SmallChange* – определяют, какими шагами будет двигаться ползунок при перемещении его пользователем (путем щелчка на полосе ниже ползунка или при нажатии на кнопку направления соответственно);
- *Orientation* - определяет расположение ползунка: вертикальное или горизонтальное;
- *Value* - главное свойство этого элемента управления. Определяет положение ползунка и то значение, которое будет возвращать этот элемент управления программе.

Элемент управления *SpinButton* (Счетчик) - это та же полоса прокрутки, лишенная самой полосы и ползунка. Все свойства, которые есть у *SpinButton*, совпадают со свойствами *ScrollBar*.

Image (Рисунок) позволяет отобразить на форме рисунок в одном из распространенных форматов, который будет реагировать на щелчок мышью или просто использоваться для украшения формы.

Главное событие элемента *Image* - событие *Click*.

Главные свойства *Image*:

- *Picture* - позволяет выбрать само изображение для формы;
- *PictureAlignment* - позволяет выбрать местонахождение изображения в отведенной ему области. По умолчанию - по центру;

- *PictureTiling* – определяет, размножать ли маленький рисунок, чтобы он покрыл все отведенную ему область.

Стандартные элементы управления изначально помещены в *ToolBox* и доступны для размещения в формах. Для размещения дополнительных элементов управления на форме нужно сначала вставить их в окно *ToolBox*. Для этого надо щелкнуть правой кнопкой мыши по пустому пространству в *ToolBox* и выбрать пункт *Additional Controls*.

Один из часто используемых дополнительных элементов управления, который есть практически на всех компьютерах – это *Calendar* (Календарь). Вид размещенного на форме элемента *Calendar* показан на рис. 3.4. При помощи этого элемента управления пользователю очень удобно выбирать нужную дату.

Главное свойство элемента *Calendar* - *Value*, то есть та дата, которая выбрана пользователем. Остальные свойства предназначены для оформления внешнего вида.

.

Рис. 3.4 Вид элемент управления *Calendar* на форме

Лабораторная работа № 4

Отладка программ в редакторе VBA

Цель работы: изучить работу с окнами *Immediate* и *Locals* при отладке программ в редакторе VBA.

Задание 1 ***Работа с окном Immediate***

1. Создайте в Excel новую рабочую книгу и сохраните ее с именем *Debug1.xlsm*.

2. Войдите в редактор VBA, вставьте в окно редактора новую форму (команда *Insert>UserForm*), разместите на форме элементы управления *Label, TextBox* и *CommandButton*. Оформите внешний вид формы, как показано на рис. 4.1.

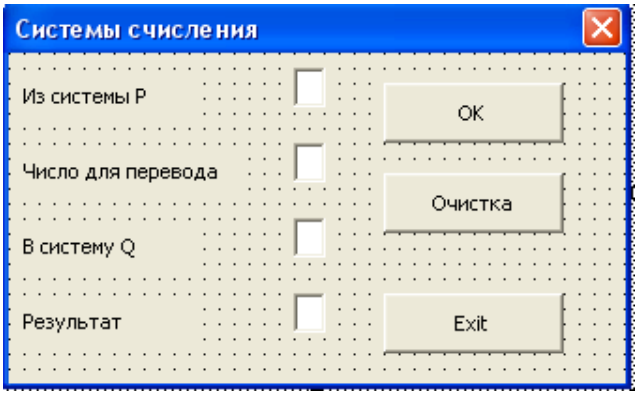


Рис. 4.1 Внешний вид формы *Системы счисления*

3. В окне *Properties* в поле *Name* для элементов *TextBox* и *CommandButton* введите имя в соответствии с таблицей 4.1.

4. Двойным щелчком по кнопке *OK* откройте окно кода и введите между строчками *Private Sub cmd_Ok_Click()* и *End Sub* следующий код:

```
Dim num_P, num_Q, num_10, num_S
num_P = Val(txt_P)
num_Q = Val(txt_Q)
For i = 1 To Len(txt_A)
```

```
num_10 = num_10 + Val(Mid(txt_A, i, 1)) * num_P ^ (Len(txt_A) - i)
   ' Debug.Print ("num_10=" & num_10)
Next i
txt_B = ""
While num_10 <> 0
num_S = num_10 Mod num_Q
   ' Debug.Print ("num_S" & num_S)
txt_B = Mid(Str(num_S), 2, 1) + txt_B
   ' Debug.Print ("txt_B=" & txt_B)
num_10 = num_10 \ num_Q
Wend
```

Таблица 4.1 Имена элементов на форме *Системы счисления*

Элемент	*Наименование*	*Имя*
Форма	Системы счисления	UserForm1
Текстовое поле	Из системы P	txt_P
Текстовое поле	Число для перевода	txt_A
Текстовое поле	В систему Q	txt _Q
Текстовое поле	Результат	txt_B
Кнопка	OK	cmd_OK
Кнопка	Очистка	cmd_Clear
Кнопка	Exit	cmd_Exitr

5. Двойным щелчком по кнопке *Exit* откройте окно кода и введите между строчками *Private Sub cmd_Exit_Click()* и *End Sub* следующий код:

```
UserForm1.Hide
```

6. Двойным щелчком по кнопке *Очистка* откройте окно кода и введите между строчками *Private Sub cmd_Clear_Click()* и *End Sub* следующий код:

```
txt_A = " "
txt_B = " "
txt_P = " "
txt_Q = " "
```

7. Запустите программу на выполнение, введите исходные данные в первые три текстовых поля и нажмите *ОК*. При отсутствии ошибок в программе в текстовом поле *Результат* будет выведен результат перевода введенного числа в системе с основанием P в систему с основанием Q.

8. Удалите символ комментария в строках процедуры *cmd_Ok_Click,* содержащих оператор *Debug.Print* для его исполнения в ходе выполнения программы. Этот оператор будет выводить текущие значения переменных *num_10*, *num_S* и текстового поля *txt_B* в окно *Immediate.*

9. Откройте окно *Immediate* (<Ctrl+G>) и проанализируйте значения, которые контролируемые величины принимали в ходе выполнения программы.

10.Оставьте окно *Immediate* видимым на экране и запустите программу командой *Step Into* в меню *Debbug*.

11. Введите исходные данные в текстовые поля формы и последовательным нажатием клавиши <F8> выполните программу в пошаговом режиме, наблюдая в окне *Immediate* за изменением значений выводимых оператором *Debug.Print* величин.

Задание 2 ***Работа с окном Locals***

1. Откройте окно *Locals* командой *Locals Window* в меню Viev.

2. Запустите программу в пошаговом режиме и проследите значения переменных *num_P, num_Q, num_10, num_S* на каждом шаге выполнения программы в окне *Locals*. Вид окна *Locals* на первом шаге выполнения цикла *For i = 1 ...Next I* показан рис. 4.2.

3. Просмотрите и попробуйте изменить в окне *Locals* свойства кнопок и текстовых полей, открыв контейнер специального объекта *ME* и контейнер соответствующего элемента управления. Вид окна *Locals* при открытых свойствах формы и кнопки *cmd_Clear* показан на рис. 4.3.

4. Сохраните книгу *Debug1.xlsm* и подготовьте отчет по лабораторной работе.

Задание для самостоятельной работы

Изучить приемы использования окна контролируемых выражений *Watches* при отладке программ в окне редактора VBA.

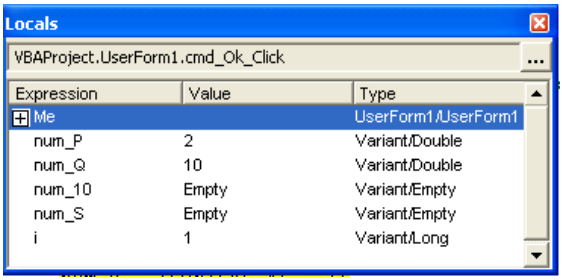


Рис. 4.2 Вид окна *Locals* на одном из шагов отладки программы

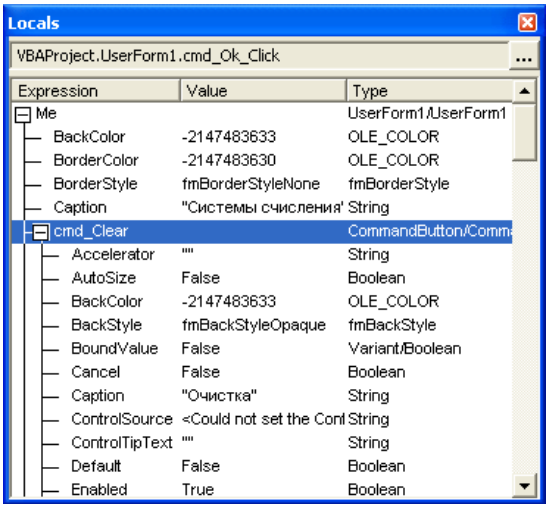


Рис. 4.3 Вид окна *Locals* при открытых свойствах формы и кнопки *cmd_Clear*

Методические материалы

Опыт показывает, что при написании программного кода могут быть допущены ошибки. Задача программиста - найти эти ошибки и устранить их до передачи программы конечному пользователю.

Все ошибки можно разделить на три большие группы - *синтаксические, логические* и *ошибки времени выполнения*.

Один из самых важных приемов в ходе отладки программы - переход в режим остановки выполнения программы, чтобы просмотреть значения переменных и вмешаться в ход выполнения программы вручную.

В режим паузы программу можно перевести:

- сразу запустить программу в режиме пошагового выполнения (меню *Debug>Step Into* или клавиша <F8>). В этом случае программа будет переходить в режим паузы после выполнения каждого оператора;
- установить в программе точку останова (*breakpoint*). Это можно сделать установкой указателя в нужной строке кода и выбором в меню *Debug* команды *Toggle Breakpoint* (<F9>). Второй вариант - просто щелкнуть мышкой по рамке слева от строки. Снятие точки останова - сделать все то же самое еще раз;
- еще одна возможность приостановить выполнение программы - воспользоваться контролируемым выражением в окне *Watches*.

В любом случае выполнение будет приостановлено в выбранном месте программы и следующий оператор, который должен быть выполнен, будет выделен желтым цветом.

В режим паузы можно:

- продолжить выполнение программы в пошаговом режиме - команда *Debug->Step Into* (<F8>);
- если в строке программы происходит вызов какой-то процедуры, которая уже отлажена, можно без остановок перейти к следующему оператору - *Debug>Step Over* (<Shift>+<F8>);
- довести выполнение начатой процедуры до конца - команда *Debug>Step Out* (Ctrl>+<Shift>+<F8>);
- исполнить код не пошагово, а участками. Для этого надо щелкнуть правой кнопкой мыши по нужному участку кода и в контекстном меню выбрать *Run to Cursor* или воспользоваться той же командой в меню *Debug* (<Ctrl>+<F8>);
- "перепрыгнуть" через какой-то участок кода, вызывающий ошибку - команда *Debug>Set Next Statement* (<Ctrl>+<F9>), а затем перетащить желтую отметку по левой границе вниз или вверх;
- вернуться к месту остановки без долгих розысков - *Show Next Statement*;
- продолжить выполнение после остановки - <F5> или воспользоваться командой *Continue* (она появится вместо команды *Run*) в меню *Run;*
- прекратить выполнение программы командой *Reset* (<Alt>+<F4>).

В окне *Immediate* (вызов из меню *View* или <Ctrl>+<G>) можно просматривать или изменять значения переменных и свойств объектов.

Вывод в окно *Immediate* значений переменной *a* можно произвести, вписав в код программы строку *Debug.Print a*.

Изменение значений перемененных и свойств в окне *Immediate* производится точно так же, как в коде программы. В окне *Immediate* можно также вызывать процедуры и функции программы или методы объектов - точно так же, как в коде программы.

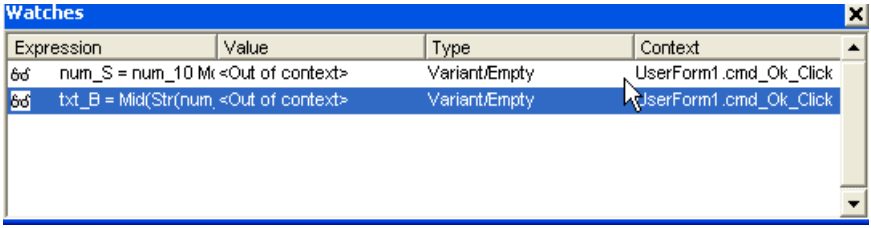
Чтобы не печатать в окне *Immediate* выражения и имена переменных, которые уже есть в коде, можно воспользоваться перетаскиванием участков кода в окно *Immediate* из окна редактора кода с нажатой клавишей <Ctrl> для копирования.

Окно *Watches* (Контролируемые выражения) предназначено для контроля за отдельными выражениями и значениями переменных.

Чтобы добавить контролируемое выражение в окно *Watches*, нужно щелкнуть по нему правой кнопкой мыши и выбрать из контекстного меню команду *Add Watch.*

Для редактирования контролируемого выражения в окне *Watches* используется команда *Edit Watch, д*ля удаления - *Delete Watch*, для добавления нового - *Add Watch.*

Вид окна *Watches* с введенными двумя контролируемыми выражениями приведен на рис. 4.4.



Ртс. 4.4 Вид окна *Watches* с добавленными контролируемыми выражениями

Лабораторная работа № 5

Работа с управляющими конструкциями в VBA

Цель работы: научиться использовать в программах VBA управляющие конструкции для программирования разветвлений и повторяющихся вычислений.

Задание 1 ***Использование оператора If… Then... Else***

1. Откройте приложение Excel и создайте новую книгу. Сохраните ее с именем *If_For.xlsm*.
2. Войдите в редактор VBA и вставьте новую форму.
3. Разместите на форме элементы управления *Label*, *TextBox* и *CommandButton* с окна *ToolBox* и отредактируйте их свойства, как показано на рис. 5.1.
4. Введите имена формы и элементов управления *TextBox* и *CommandButton* в соответствии с таблицей 5.1.
5. Двойным щелчком на кнопке *Выполнить* откройте окно кода и введите следующий код:

```
Private Sub cmd_OK1_Click()
Dim a As Single, b As Single, c As Single, res As Single
a = Val(txt_a)
b = Val(txt_b)
c = Val(txt_c)
If a > b And b > c Then
MsgBox ("Введены значения a > b и b > c")
res = a + b + c
txt_Res = Str(Res)
ElseIf (a < b Or b < c) Then
MsgBox ("Введены a < b и b < c")
res = a + b - c
txt_Res = Str(res)
Else
MsgBox "Условия не выполнены"
res = 0
txt_Res = Str(res)
End If
End Sub
```

Рис. 5.1 Вид формы в режиме конструктора

Таблица 5.1 Имена элементов на форме *Операторы ветвления*

Элемент	*Наименование*	*Имя*
Форма	Операторы ветвления	frm_If
Текстовое поле	Введите a	txt_a
Текстовое поле	Введите b	txt_b
Текстовое поле	Введите c	txt _c
Текстовое поле	Результат	txt_Res
Кнопка	OK	cmd_OK1
Кнопка	Очистить	cmd_Clear1
Кнопка	Выход	cmd_Exit1

6. По аналогии с п. 5 для кнопки *Очистить* введите следующий код для очистки текстовых полей формы:

Private Sub cmd_Clear1_Click()
txt_a=" " : txt_b=" ": txt_c=" ": txt_Res=" "
End Sub

7. Аналогично для кнопки *Выход* введите код:

Private Sub_cmd_Clear1_Click()
frm_If.Hide
End Sub

8. Протестируйте разработанную программу, выполняя ее при разных вариантах исходных данных.

9. Запустите программу в режиме отладки и используя окно *Locals,* просмотрите текущие значения переменных *a, b, c* и *res* и последовательность выполнения операторов в программе в зависимости от значений исходных данных.

10.Сохраните разработанную форму.

Задание2 ***Использование OptionButton***

1. Вставьте в проект новую форму и оформите ее, как показано на рис. 5.2. Имена элементов управления *TextBox* и *CommandButton* введите по аналогии с заданием 1, имена *OptionButton* оставьте по умолчанию.

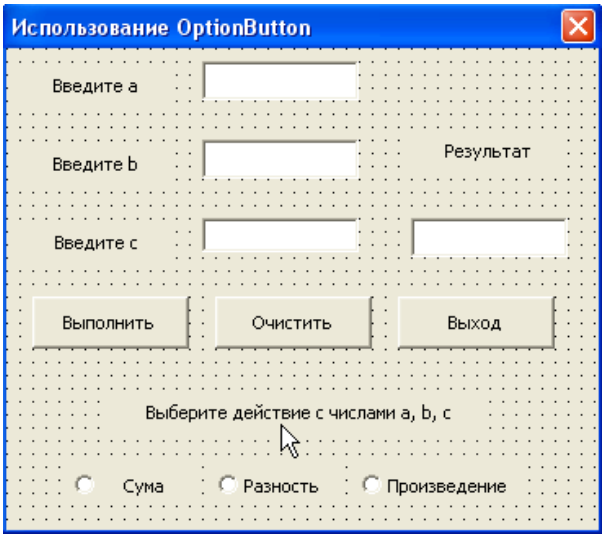


Рис. 5.2 Вид формы в режиме конструктора

2. Для кнопки *Выполнить* введите следующий код:

```
Private Sub cmd_OK1_Click()
Dim a As Single, b As Single, c As Single, res As Single
a = Val(txt_a): b = Val(txt_b): c = Val(txt_c)
If OptionButton1 = True Then
MsgBox ("Операция a+b+c")
```

res = a + b + c
txt_res = Str(res)
ElseIf OptionButton2 = True Then
MsgBox ("Операция a-b-c")
res = a - b - c
txt_res = Str(res)
ElseIf OptionButton3 = True Then
*MsgBox " Операция a*b*c"*
*res = a * b * c*
txt_res = Str(res)
Else
MsgBox ("Операция не выбрана")
End If
Sub End

3. Введите код отклика на нажатие кнопок *Очистить* и *Выход* по аналогии с п. п 6 - 7 задания 1.

4. Протестируйте созданную программу и сохраните разработанную форму с именем *Form_OB.frm*.

Задание 3 Использование операторов цикла *For…Next*

1. В книге *If_For.xlsm* вставьте новый модуль и создайте в нем процедуру с именем *For_max* и опцией *Private*.

2. Вставьте между строками заголовка и окончания процедуры *For_max* следующий код:

Dim i As Integer
Dim n As Integer
Dim x As Single, max As Single
max = 0
n = InputBox ("Введите количество чисел")
For i = 1 To n
x = InputBox ("Введите" & Str(i) & "-е число")
If x > max Or i = 1 Then max = x
Next i
MsgBox "max = " & Str(max)

3. Протестируйте процедуру *For_max*. Обратите внимание на конструкцию *For…Next* .

4. Вставьте по аналогии с п. 1 новую процедуру с именем *For_min*. Скопируйте в нее код предыдущей процедуры и измените

его так, чтобы определялось минимальное число из введенных чисел.

5. Разработайте форму для выбора одной из двух процедур *For_max* или *For_min* с использованием элементов *OptionButton*.

6. Протестируйте и сохраните созданную форму с именем *max_min.frm*.

Задание 4 Работа с конструкциями While…Loop, Until…Loop

1. Вставьте новую форму и оформите ее, показано на рис. 5.3.

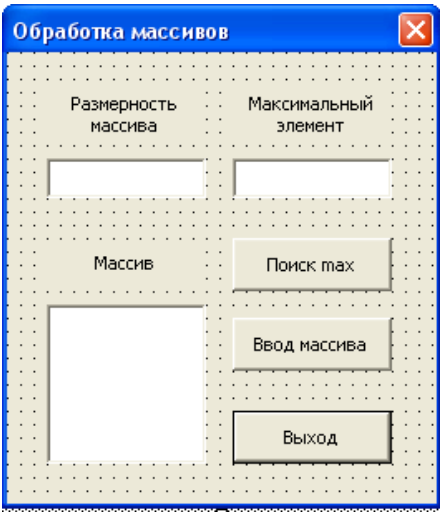


Рис. 5.3 Вид формы в режиме конструктора

2. Имена элементам *TextBos*, *ListBox* и *CommandButton* присвойте в соответствии с таблицей 5.2.

3. В разделе объявлений модуля объявите динамический массив и переменую для хранения его размерности при переопределении:

Option Base 1

Dim Massiv() As Single, Size As Integer

Таблица 5.2 Имена элементов на форме *Обработка массивов*

Элемент	*Наименование*	*Имя*
Форма	Обработка массивов	frm_mas
Текстовое поле	Размерность массива	txt_razm
Текстовое поле	Максимальный элемент	txt_max
Кнопка	Ввод массива	cmd_Vvod
Кнопка	Поиск max	cmd_max
Кнопка	Выход	cmd_Exit1
Список	Массив	lst_Massiv

4. Для кнопки *Ввод массива* введите следующий код:

```
Private Sub cmd_Vvod_Click()
Dim i As Integer
Size = Val(txt_razm.Text)
ReDim Massiv(Size)
i = 1
Do While i <= Size
  Massiv(i) = Val(InputBox("Массив(" + Str(i) + ")=", "Ввод массива"))
  lst_Massiv.AddItem "A(" + Str(i) + ")=" + Str(Massiv(i))
i = i + 1
Loop
End Sub
```

5. Для кнопки *Поиск max* введите следующий код обработчика:

```
Private Sub cmd_max_Click()
Dim i As Integer
max = Massiv(1)
i = 1
Do Until i > Size
   If Massiv(i) > max Then max = Massiv(i)
   i = i + 1
Loop
txt_max.Text = Str(max)
End Sub
```

6. Для кнопки *Выход* введите код для закрытия формы:

Private Sub cmd_Exit_Click()

frm_mas.Hide

End Sub

7. Протестируйте программу. Обратите внимание на конструкции *While…Loop, Until…Loop*, опцию *Option Base 1*, на способ заполнения позиций элемента *ListBox*.

8. Добавьте на форму кнопку *Очистить* для очистки текстовых полей формы. Напишите и введите для нее код. Протестируйте работу кнопки *Очистить*.

9. Сохраните форму *frm_mas* и закройте книгу Excel с сохранением изменений. Оформите отчет по лабораторной работе.

Задание для самостоятельной работы

Разработайте проект, включающий форму и необходимые программы для поиска макимального и минимального элемента в двумерном динамическом массиве.

Методические материалы

Операторы ветвления - одни из самых важных и часто используемых конструкций в языках программирования. Общий принцип их работы прост: проверяется соответствие каким-то условиям (истинность или ложность выражений) и в зависимости от этого выполнение программы направляется по одной или другой ветви.

Оператор *If…Then... Else* применяется:

- когда нужно проверить на соответствие одному условию и в случае соответствия сделать какое-то действие;

If nTemperature < 10 Then

MsgBox "Одеть куртку"

End If

- когда нужно сделать то же, что и в предыдущем примере, а в случае несоответствия выполнить другое действие;

If nTemperature < 10 Then

MsgBox "Одеть куртку"

Else

MsgBox "Одеть ветровку"

End If

- когда нужно проверить на соответствие нескольким условиям;

If (nTemperature < 10) And (bRain = True) Then
MsgBox "Одеть куртку и взять зонтик"
End If

- в случае, если первая проверка вернула *False*, нужно проверить на соответствие еще нескольким условиям (в этом случае удобно использовать *ElseIf*);

If (bIGoInCar = True) Then
MsgBox "Одеться для машины"
ElseIf nTemperature < 10 Then
MsgBox "Одеть куртку"
Else
MsgBox "Можно идти в рубашке"
End If

Оператор *Select Case* идеально подходит для проверки одного и того же значения, которое нужно много раз сравнить с разными выражениями;

Select Case sDayOfWeek
Case "Понедельник"
MsgBox "Салат из шпината"
Case "Вторник"
MsgBox "Салат из морской капусты"
Case Else
MsgBox "На этот день у нас ничего не предусмотрено"
End Select

Операторы цикла в VBA используются в ситуациях, когда нужно выполнить какое-либо действие несколько раз. Если известно, сколько раз нужно выполнить действие, используется конструкция *For…Next*:

For iCounter = 1 to 10
MsgBox "Счетчик: " & iCounter
Next

Чтобы указать, насколько должно прирастать значение счетчика, используется ключевое слово *Step*:

For iCounter = 1 to 10 Step 2
MsgBox "Счетчик: " & iCounter
Next

Можно и уменьшать исходное значение счетчика:

For iCounter = 10 to 1 Step -2
MsgBox "Счетчик: " & iCounter
Next

Для безусловного выхода из конструкции *For…Next* используется команда *Exit For*.

VStop = InputBox ("Введите значение останова")
VInput = CInt(VStop)
For iCounter = 1 to 10
MsgBox "Счетчик: " & iCounter
If iCounter =VInput Then Exit For
Next

Очень часто в VBA требуется сделать какое-нибудь действие со всеми элементам коллекции или массива - перебрать все открытые документы, все листы Excel, все ячейки в определенном диапазоне и т.п. Для того, чтобы пройти циклом по всем элементам коллекции, используется команда *For Each … Next*:

For Each oWbk in Workbooks
MsgBox oWbk.Name
Next

При использовании этого приема можно очень просто найти и получить ссылку на нужный нам объект:

For Each oWbk in Workbooks
If oWbk.Name = "Сводка.xls" Then
Set oMyWorkBook = oWbk
Exit For
End If
Next

Когда число повторений зависит от какого-либо условия, используются конструкции *Do While…Loop и Do Until…Loop.*

Конструкция *Do While* означает: выполнять какое-либо действие до тех пор, пока условие истинно:

Do While MyVar < 10
MyVar = MyVar + 1 : MsgBox " MyVar = " & MyVar
Loop

Второй вариант - *Do Until:* цикл будет выполняться до тех пор, пока условие ложно.

Do Until MyVar >= 10

MyVar = MyVar + 1
MsgBox " MyVar = " & MyVar
Loop

Можно переписать цикл так, чтобы условие проверялось после завершения цикла:

Do
MyVar = MyVar + 1
WScript.Echo "MyVar = " & MyVar
Loop While MyVar < 10

В этом случае цикл будет выполнен по крайней мере один раз.

Немедленный выход из цикла можно произвести по команде *Exit Do*.

Массивы используются для хранения в памяти множества значений. Объявление массива производится очень просто:

Dim MyArray (10) As Integer

Параметр 10 - это верхняя граница массива. Количество элементов, которое может хранить массив, от 0 до верхней границы включительно. Если требуется, чтобы нумерация элементов в массиве начиналась с 1, в разделе объявлений модуля используется команда *Option Base 1*

Массивы могут быть многомерными:

Dim MyArray (4, 9)

В каждой строке многомерного массива удобно хранить данные, относящиеся к одному объекту (например, имя сотрудника, уникальный номер, номер телефона). В VBA в одном массиве может быть до 60 измерений.

В динамических массивах размер можно изменять в ходе выполнения программы. Динамический массив объявляется следующим образом:

Dim MyArray () '- объявляем массив без верхней границы
ReDim MyArray (4) ' - изменяем размер массива

Чтобы старые значения сохранить, используется ключевое слово *Preserve*:

ReDim Preserve MyArray (7)

Массивы можно создавать и заполнять одновременно при помощи встроенной функции *Array()*:

Dim MyArray
MyArray = Array(100, 200, 300, 400, 500)

Очистить массив можно командой *Erase*:

Erase MyArray

Массив фиксированной длины просто очищается, динамический массив разинициализируется - его придется инициализировать (определять размер) заново.

В динамических массивах часто неизвестно, сколько элементов находится в массиве. Для определения количества элементов используется функция *UBound()*.

При программировании в VBA вместо массивов в объектных моделях приложений Microsoft Office обычно используются коллекции. Коллекции - это специальные объекты, которые предназначены для хранения наборов одинаковых элементов. Например, в Word предусмотрена коллекция *Documents* для хранения элементов *Document* - то есть всех открытых документов, в Excel - коллекции *Workbooks* (открытые книги) и *Worksheets* (листы в книге) и т.п. Коллекции обычно удобнее, чем массивы: они изначально безразмерны и в них предусмотрен стандартный набор свойств и методов - метод *Add()* для добавления нового элемента, свойство *Count* для получения информации о количестве элементов, метод *Item()* для получения ссылки на нужный элемент.

Лабораторная работа № 6

Обработка табличных Excel данных в VBA

Цель работы: научиться обрабатывать табличные данные Excel в среде VBA.

Задание 1 ***Ввод данных в таблицу Excel***

1. Откройте приложение Excel и создайте новую книгу. Сохраните ее с именем *VBA*_Excel.*xlsm*.

2. Откройте окно редактора VBA и вставьте новую форму для оформления шапки таблицы. Расположите на ней элементы управления и оформите их внешний вид, как показано на рис. 6.1.

3. Введите имя формы и имена командных кнопок в соответствии с таблицей 6.1.

Рис. 6.1 Вид формы *Заголовок таблицы*

Таблица 6.1 Имена элементов на форме *Заголовок таблицы*

Элемент	*Наименование*	*Имя*
Форма	Заголовок таблицы	frm_Fut
Кнопка	Ввод	cmd_Vvod
Кнопка	Выход	cmd_Exit

4. В окне кода в разделе формы *frm_Fut* введите следующий код:

```
Private Sub cmd_Vvod_Click()
'Формирование шапки таблицы на 1-м листе
 Sheets(1).Activate
 With Range("A1:H1")
```

```
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
        .ReadingOrder = xlContext
        .MergeCells = True
    End With
 Cells(1, 1) = "Расчет стипендии студентам энергофака"
Cells(2, 1) = "Группа"
Cells(2, 2) = "Ф.И.О."
Cells(2, 3) = "Экз.1"
Cells(2, 4) = "Экз.2"
Cells(2, 5) = "Экз.3"
Cells(2, 6) = "Экз.4"
Cells(2, 7) = "Экз.5"
Cells(2, 8) = "Стипендия"
With Range("A1:H1000")
    .HorizontalAlignment = xlCenter
    .VerticalAlignment = xlCenter
    .ReadingOrder = xlContext
 End With
  cmd_Vvod.Enabled = False
End Sub

Private Sub cmd_Exit_Click()
frm_Fut.Hide
End Sub
```

5. Запустите на выполнение форму *frm_Fut* и просмотрите результат на *Лист1* книги Excel. Возможный вариант шапки таблицы показан на рис. 6.2. При необходимости вы можете внести изменения в заголовки столбцов таблицы или оформить ее на другом листе.

6. Обязательно сохраните форму *frm_Fut* в личную папку.

	A	B	C	D	E	F	G	H
1	Расчет стипендии студентам энергофака							
2	Группа	Ф.И.О.	Экз.1	Экз.2	Экз.3	Экз.4	Экз.5	Стипендия

Рис. 6.2 Вид шапки таблицы

7. Вставьте новую форму для ввода данных. Расположите на ней элементы управления и оформите их внешний вид, как показано на рис. 6.2.

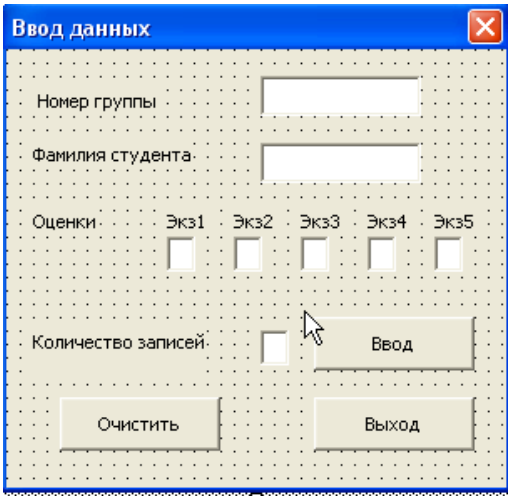


Рис. 6.2 Вид формы *Ввод данных* в режиме конструктора

8. Введите имя формы и имена элементов управления в соответствии с таблицей 6.2.

9. В окне редактора кода в разделе формы *frm_Vvod* введите следующий код:

```
Dim i As Double
Private Sub cmd_Vvod_Click()
Cells(i, 1) = txt_Ind.Text
Cells(i, 2) = txt_FIO.Text
Cells(i, 3) = CInt(txt_M1.Text)
Cells(i, 4) = CInt(txt_M2.Text)
Cells(i, 5) = CInt(txt_M3.Text)
Cells(i, 6) = CInt(txt_M4.Text)
Cells(i, 7) = CInt(txt_M5.Text)
txt_N.Enabled = True
txt_N.Text = CStr(i)
txt_N.Enabled = False
i = i + 1
```

```
End Sub
Private Sub cmd_Clear_Click()
txt_Ind.Text = " ": txt_FIO.Text = " " : txt_M1.Text = " "
txt_M2.Text = " ": txt_M3.Text = " ": txt_M4.Text = " "
txt_M5.Text = " "
End Sub
Private Sub cmd_Exit_Click()
frm_Vvod.Hide
End Sub
Private Sub UserForm_Initialize()
  i = 1
  Do While Cells(i, 1) > " "
    i = i + 1
  Loop
  txt_N.Enabled = True
  txt_N.Text = CStr(i - 1)
  txt_N.Enabled = False
End Sub
```

Таблица 6.2 Имена элементов на форме *Ввод данных*

Элемент	*Наименование*	*Имя*
Форма	Ввод данных	Frm_Vvod
Текстовое поле	Группа	txt_Ind
Текстовое поле	Количество запи-	txt_N
Текстовое поле	Фамилия студента	txt_FIO
Текстовое поле	Экз.1	txt_M1
Текстовое поле	Экз.2	txt_M2
Текстовое поле	Экз.3	txt_M3
Текстовое поле	Экз.4	txt_M4
Текстовое поле	Экз.5	txt_M5
Кнопка	Ввод	cmd_Vvod
Кнопка	Очистить	cmd_Clear
Кнопка	Выход	cmd_Exit

10. Запустите *frm_Vvod* и заполните данными не менее 10 строк таблицы *Расчет стипендии.*

11.Сохраните форму *frm_Vvod*.

Задание 2 ***Организация вычислений в таблице***

1. Вставьте новую форму для расчета стипендии. Расположите на ней элементы управления и оформите их внешний вид, как показано на рис. 6.3

.

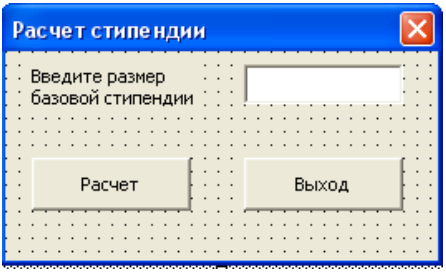


Рис. 6.3 Вид формы *Расчет стипендии*

2. Введите имя формы и имена командных кнопок в соответствии с таблицей 6.3.

Таблица 6.3 Имена элементов на форме *Расчет стипендии*

Элемент	*Наименование*	*Имя*
Форма	Заголовок таблицы	*frm_Ras*
Текстовое поле	Введите размер базовой стипендии	*txt_St*
Кнопка	Расчет	*cmd_Ras*
Кнопка	Выход	*cmd_Exit*

3. В окне кода в разделе формы *frm_Ras* введите следующий код:

```
Dim Sb As Double
Dim Stip As Double
Private Sub cmd_Exit_Click()
frm_Ras.Hide
End Sub
Private Sub cmd_Ras_Click()
St = CSng(txt_St)
```

```
i = 3
Do While Sheets(1).Cells(i, 2) > " "
k0 = 0: k1 = 0: k3 = 0: k4 = 0: k5 = 0: k6 = 0: k7 = 0
k8 = 0: k9 = 0: k10 = 0: k2 = 0
For j = 1 To 5
If Cells(i, j + 2) = 0 Then k0 = k0 + 1
If Cells(i, j + 2) = 1 Then k1 = k1 + 1
If Cells(i, j + 2) = 2 Then k2 = k2 + 1
If Cells(i, j + 2) = 3 Then k3 = k3 + 1
If Cells(i, j + 2) = 4 Then k4 = k4 + 1
If Cells(i, j + 2) = 5 Then k5 = k5 + 1
If Cells(i, j + 2) = 6 Then k6 = k6 + 1
If Cells(i, j + 2) = 7 Then k7 = k7 + 1
If Cells(i, j + 2) = 8 Then k8 = k8 + 1
If Cells(i, j + 2) = 9 Then k9 = k9 + 1
If Cells(i, j + 2) = 10 Then k10 = k10 + 1
Next j
Sb = (k1 * 1 + k2 * 2 + k3 * 3 + k4 * 4 + k5 * 5 _
+ k6 * 6 + k7 * 7 + k8 * 8 + k9 * 9 + k10 * 10) / 5
If (k0 + k1 + k2 + k3) > 0 Then
  Stip = 0
       Else
 If Sb >= 9 Then
  Stip = St * 1.6
       Else
   If (Sb >= 8 And Sb < 9) Then
  Stip = St * 1.4
       Else
    If (Sb >= 6 And Sb < 8) Then
  Stip = St * 1.2
   End If
  End If
 End If
End If
   Cells(i, 8) = Stip
i = i + 1
Loop
End Sub
```

4. Запустите на выполнение форму *frm_Ras*. В результате на *Лист1* будет заполнен столбец *Стипендия* значениями вычисленной стипендии. Возможный вариант вида заполненной таблицы приведен на рис. 6.4.

5. Проанализируйте результаты работы и коды разработанных программ. Установите для себя однозначное соответствие введенных данных в каждой ячейке таблицы на листе Excel и строк программного кода созданных форм.

6. Сохраните форму *frm_Ras*.

	A	B	C	D	E	F	G	H
1	Расчет стипендии студентам энергофака							
2	Группа	Ф.И.О.	Экз.1	Экз.2	Экз.3	Экз.4	Экз.5	Стипендия
3	106113	Егоров	5	6	7	8	7	686400
4	106313	Белый	10	8	8	8	9	800800
5	106113	Синявин	9	8	8	9	9	800800
6	106213	Горностай	9	8	10	9	9	915200
7								

Рис. 6.4 Вариант таблицы расчета стипендии

Задание 3 ***Сортировка данных в таблице***

1. Вставьте новую форму для сортировки данных в таблице.

2. Выполните для нее действия, аналогичные п. п. 1, 2 предыдущего задания. Внешний вид новой формы показан на рис. 6.5, имена объектов на форме – в таблице 6.4.

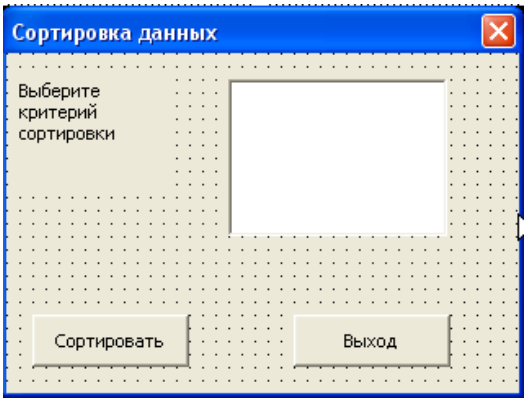


Рис. 6.5 Внешний вид формы *Сортировка данных*

Таблица 6.4 Имена элементов на форме *Сортировка данных*

Элемент	*Наименование*	*Имя*
Форма	Сортировка данных	frm_Sort
Кнопка	Сортировать	cmd_Sort
Кнопка	Выход	cmd_Exit

3. В окне кода в разделе формы *frm_Sort* введите следующий код:

```
Private Sub UserForm_Activate()
' Заполнение списка ListBox1
ListBox1.AddItem "Группа"
ListBox1.AddItem "ФИО"
ListBox1.AddItem "Экз.1"
ListBox1.AddItem " Экз.2"
ListBox1.AddItem " Экз.3"
ListBox1.AddItem "Экз.4"
ListBox1.AddItem "Экз.5"
End Sub
Private Sub cmd_Sort_Click()
' Определение количества строк в таблице
n = 3
Do While Cells(n, 1) > " "
  n = n + 1
Loop
n = n - 1
'пример выбранного критерия
k = ListBox1.ListIndex + 1
' Сортировка
i = 3
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 " & ListBox1.Text & " завершена!", vbCritical, _ "Сортировка"
End Sub
Private Sub cmd_Exit_Click()
frm_Sort.Hide
End Sub
```

4. Проверьте выполнение сортировки данных в таблице по выбранному критерию. Проанализируйте код каждой процедуры, будьте готовы пояснить назначение каждой строки кода программы.

5. Сохраните форму *frm_Sort*.

Задание 4 ***Формирование ведомости по данным таблицы***

1. Вставьте новую форму для формирования ведомости выплаты стипендии. Внешний вид новой формы показан на рис. 6.6, имена объектов на форме – в таблице 6.5.

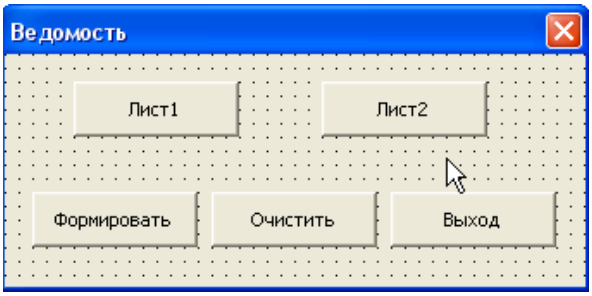


Рис. 6.6 Внешний вид формы *Ведомость*

Таблица 6.4 Имена элементов на форме *Ведомость*

Элемент	*Наименование*	*Имя*
Форма	Ведомость	frm_L2
Кнопка	Лист1	cmd_L1
Кнопка	Лист2	cmd_L2
Кнопка	Формировать	cmd_F
Кнопка	Очистить	cmd_Clear
Кнопка	Выход	cmd_Exit

2. В окне кода в разделе формы *frm_L2* введите следующий код:

```
Private Sub cmd_F_Click()
Sheets(1).Activate
' Формирование шапки таблицы на втором листе
With Sheets(2).Range("A1:E1")
      .HorizontalAlignment = xlCenter
      .VerticalAlignment = xlCenter
      .ReadingOrder = xlContext
      .MergeCells = True
 End With
 With Sheets(2).Range("A3:G100")
      .HorizontalAlignment = xlCenter
      .VerticalAlignment = xlCenter
      .ReadingOrder = xlContext
 End With
Sheets(2).Cells(1, 1) = "Ведомость выдачи стипендии"
Sheets(2).Cells(2, 2) = "ФИО"
Sheets(2).Cells(2, 3) = "Сумма"
'i- номер строки на  1-м листе; k – на 2-м листе
i = 3
k = 3
Do While Cells(i, 2) > ""
  If Cells(i, 8) > 0 Then
  Sheets(2).Cells(k, 2) = Sheets(1).Cells(i, 2)
  Sheets(2).Cells(k, 3) = Sheets(1).Cells(i, 8)
    k = k + 1
End If
```

```
i = i + 1
Loop
Sheets(2).Activate
Range("A20").Select
End Sub
Private Sub cmd_L1_Click()
Sheets(1).Activate
End Sub
Private Sub cmd_L2_Click()
Sheets(2).Activate
End Sub
Private Sub cmd_Clear_Click()
Sheets(2).Activate
Range("A1:Z100").Clear
Sheets(1).Activate
End Sub
Private Sub cmd_Exit_Click()
Sheets(1).Activate
frm_L2.Hide
End Sub
```

3. Протестируйте и сохраните форму *frm_L2*.

4. Подготовьте отчет по лабораторной работе и закройте книгу *VBA_Excel.xlsm* с сохранением изменений.

Задание для самостоятельной работы

1. Подробно разберите каждую процедуру и дополните код комментариями, поясняющими все операторы ветвления и цикла.

2. Опишите словесно алгоритм сортировки данных по выбранному столбцу. Поясните, как достигается при сортировке целостность данных в таблице.

Методические материалы

Наиболее часто используемый объект при работе с таблицами Excel - это объект *Range*. Этот объект может представлять одну ячейку, несколько ячеек (в том числе несмежные ячейки или наборы несмежных ячеек) или целый лист.

Самый простой способ получить объект *Range* - воспользоваться свойством *Range*. Это свойство предусмотрено для объектов *Application*, *Worksheet* и самого объекта *Range* (если нужно создать новый диапазон на основе уже существующего).

Например, получить ссылку на объект *Range*, представляющий ячейку *A1*, можно так:

Dim oRange As Range

Set oRange = Worksheets("Лист1").Range("A1").

Код для получения ссылки на диапазон ячеек A1 : D10 может быть таким:

Dim oRange As Range

Set oRange = Worksheets("Лист1").Range("A1:D10").

Второй способ - воспользоваться свойством *Cells*. Возможностей у этого свойства меньше - мы можем вернуть диапазон, состоящий только из одной ячейки. Зато мы можем использовать более удобный синтаксис для передачи переменных, перехода в любую сторону на любое количество ячеек и т.п. Например, для получения ссылки на ячейку *D1* можно использовать код вида:

Dim oRange As Range

Set oRange = Worksheets("Лист1").Cells(1, 4).

Чтобы получить диапазон, состоящий из нескольких ячеек, удобно применять свойства *Range* и *Cells* вместе:

Dim oRange

Set oRange = Range(Cells(1, 1), Cells(5, 3)).

Третий способ - воспользоваться многочисленными свойствами объекта *Range*, которые позволяют изменить текущий диапазон или создать на основе его новый.

Обычно после того, как нужная ячейка найдена, в нее нужно что-то записать. Для этой цели используется свойство *Value*, например:

oRange.Value = "Мое значение" .

Важные свойства объекта *Range*:

- *Address* - позволяет вернуть адрес текущего диапазона;
- *Cells* –позволяет сделать ссылку на отдельную ячейку диапазона;
- *Count* - возвращает количество ячеек в диапазоне;
- *Interior* - позволяет установить цвет ячейки диапазона;
- *Item* - позволяет получить еще один объект *Range*, который определяется путем смещения исходного диапазона;

- *Name* - возможность получить ссылку на специальный объект именованного диапазона *Name;*
- *Orientation* - позволяет сориентировать текст в ячейках. Указывается угол наклона в градусах;
- *Range* - позволяет создать новый диапазон на основе уже существующего;
- *Value* - позволяет получить или назначить значение (числовое, текстовое или какое-либо другое) ячейкам диапазона;

Наиболее важные методы объекта *Range*:

- *Activate()* - выделяет текущий диапазон и устанавливает курсор ввода на его первую ячейку;
- *AutoFill()* - возможность использовать автозаполнение для диапазона;
- методы *Clear…* позволяют очистить содержимое диапазона - от значений, форматирования, комментарий ;
- *Copy()* - возможность скопировать диапазон в другое место. Если место назначения не указано, он копируется в буфер обмена. Аналогично работает метод *Cut()*, при котором данные исходного диапазона вырезаются;
- *Delete()* - удаляет данные текущего диапазона;
- метод *Insert()* позволяет вставить ячейки в диапазон, сдвинув остальные с выбором направления - вправо или вниз;
- метод *Justify()* позволяет равномерно распределить текст по диапазону. Если в данный диапазон он не помещается, он будет распространен на соседние ячейки (с перезаписью их значений);
- метод *Merge()* позволяет объединить все ячейки диапазона в одну. Разбить обратно такую ячейку на несколько обычных можно при помощи метода *UnMerge();*
- *Select()* - возможность выделить указанный диапазон. Объекта *Selection* в Excel нет - вместо него есть возможность получить объект *Range*, представляющий выделенную область;
- *Show()* - экран будет проскроллирован таким образом, чтобы показать указанный диапазон;
- *Sort()* - возможность произвести сортировку ячеек в диапазоне.

Лабораторная работа № 7

Создание пользовательских приложений

Цель работы: научиться создавать пользовательские приложения для обработки данных в Excel в среде VBA.

Задание1 ***Разработать приложение в среде VBA для создания и обработки таблицы в Excel по результатам сдачи студентами экзаменационной сессии.***

Обеспечить возможность:

1. Ввода произвольного количества строк в таблицу, содержащих сведения об индексе группы, фамилии студента и оценках по каждому экзамену.
2. Вычисление стипендии студентам в зависимости от среднего балла за сессию по установленным нормативам и занесение данных расчета в созданную таблицу.
3. Сортировку данных в таблице по индексу группы, фамилии студента и размеру стипендии.
4. Формирование ведомости выдачи стипендии студентам на отдельном листе.

Рекомендуемая последовательность выполнения задания.

1. Создайте новую книгу Excel и сохраните ее с именем *VBA_Tabl.xlsm.*
2. Откройте редактор VBA и импортируйте сохраненные в личной папке формы *frm_Fut*, *frm_Vvod, frm_Ras. frm_Sort и frm_L2* из предыдущей лабораторной работы.
3. Протестируйте импортированные формы и при необходимости отредактируйте код отдельных процедур.
4. Запустите форму *frmFut* и оформите заголовок (шапку) главной таблицы.
5. Запустите форму *frmVvod* и введите не менее десяти строк данных о результатах сдачи зимней сессии студентами своего курса.
6. Рассчитайте стипендию каждому студенту в форме *frm_Ras.* Используйте действующие на момент расчета нормативы начисления стипендии в зависимости от среднего балла.

7. Проверьте работу формы *frm_Sort* по сортировке данных в таблице по заданным в задании критериям.

8. Создайте на *Лист2* ведомость выдачи стипендии студентам, используя форму *frm_L2*.

9. Вставьте новую форму в проект *VBA_Tabll* и оформите ее внешний вид, как показано на рис. 7.1.

10. Задайте имена объектам на форме и самой форме в соответствии с таблицей 7.1.

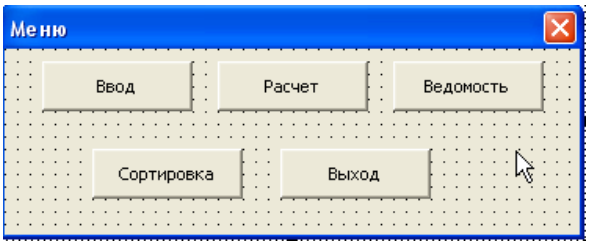


Рис. 7.1 Внешний вид формы *Меню*

Таблица 7.1 Имена элементов на форме *Меню*

Элемент	*Наименование*	*Имя*
Форма	Меню	frm_Menu
Кнопка	Ввод	cmd_Menu1
Кнопка	Расчет	cmd_Menu2
Кнопка	Ведомость	cmd_L2
Кнопка	Сортировка	cmd_Menu3
Кнопка	Выход	cmd_Exit

11. В окне кода раздела формы *frm_Menu* введите код:

```
Private Sub cmd_Menu1_Click()
Load frm_Vvod
frm_Vvod.Show
End Sub
Private Sub cmd_Menu2_Click()
Load frm_Ras
frm_Ras.Show
End Sub
```

Private Sub cmd_Menu3_Click()
Load frm_L2
frm_L2.Show
End Sub
Private Sub cmd_Menu4_Click()
Load frm_Sort
frm_Sort.Show
End Sub
Private Sub cmd_Menu5_Click()
Unload Me
End
End Sub

12. Протестируйте работу разработанного приложения. Оцените удобства работы с таблицами с использованием главного меню приложения (форма *frm_Menu*) по сравнению с вызовом каждой формы вручную.

13. Создайте кнопку с названием *Главное меню* на *Лист1* книги *VBA_Tabl.xlsm* для вызова формы *frm_Menu* созданного приложения после открытия книги.

14. Самостоятельно добавьте в код программы приложения процедуру вызова главного меню приложения при открытии книги *VBA_Tabl.xlsm*.

15. Сохраните книгу *VBA_Tabl.xlsm* и форму *frm_Menu*. Оформите отчет по лабораторной работе.

Задание для самостоятельной работы

Вариант 1

1. Создать таблицу, которая содержит сведения о сотрудниках института.

Структура таблицы:

- фамилия;
- пол;
- название отдела;
- дата рождения;
- дата поступления на работу;
- должность;
- оклад.

2. Предусмотреть возможность добавления в таблицу произвольного количества строк.

3. Рассчитать:

- стаж работы всех сотрудников;
- средний стаж работы сотрудников заданного отдела;
- количество сотрудников с окладом ниже заданного.

4. Создать на 2 листе книги таблицу, которая содержит список сотрудников пенсионного возраста (на сегодняшний день), указав стаж работы. Учесть разницу в пенсионном возрасте женщины и мужчины.

5. Сортировать данные в таблице по заданному столбцу.

6. Увеличить оклад на заданное число процентов лицам со стажем работы выше заданного.

Вариант 2

1. Создать таблицу, которая содержит сведения о заказе на ремонт оборудования.

Структура таблицы:

- номер заказа;
- дата оформления заказа;
- фамилия заказчика;
- наименование оборудования;
- сложность ремонта (низкая, средняя, высокая);
- стоимость;
- дата окончания ремонта.

2. Предусмотреть возможность добавления в таблицу произвольного количества строк.

3. Определить:

- суммарную стоимость всех заказов;
- количество заказов на ремонт заданного вида сложности;
- минимальную стоимость ремонта.

4. Создать на 2 листе книги таблицу, которая содержит сведения о продолжительности ремонта заказов, оформленных в течение заданного сезона (например, весной): номер заказа, фамилия заказчика, наименование оборудования, продолжительность заказа.

5. Сортировать данные в таблице позаданному столбцу.

6. Увеличить стоимость ремонта на X% (значение X задано) у оборудования с заданным наименованием.

Вариант 3

1. Создать таблицу, которая содержит сведения о междугородных разговорах.

Структура таблицы:

- фамилия абонента;
- номер телефона;
- дата разговора;
- код города;
- длительность разговора в минутах;
- стоимость 1 минуты.

2. Предусмотреть возможность добавления в таблицу произвольного количества строк.

3. Определить:

- максимальную стоимость разговора;
- суммарную стоимость всех разговоров;
- общее количество разговоров в город с заданным кодом.

4. Создать на 2 листе книги таблицу, которая содержит сведения о стоимости разговоров, которые состоялись в интервале между двумя заданными датами.

5. Сортировать данные в таблице по заданному столбцу.

6. Увеличить стоимость всех разговоров на X% (значение X задано).

7. Удалить сведения о разговорах абонента с заданной фамилией.

Методические материалы

Краткое описание встроенных функций VBA, которые могут быть полезными при разработке пользовательских приложений.

1. Функции преобразования типов данных:

- *Str()* - позволяет перевести числовое значение в строковое;
- *Val()* – возвращает из строковой переменной только числовое значение. При этом эта функция читает данные слева направо и останавливается на первом нечисловом значении (допускается единственное нечисловое значение – десятичная точка).

2. Строковые функции VBA:

- *ASC()* – возвращает числовой код для переданного символа.

- Chr() - возвращает символ по его числовому коду;
- *InStr()* и *InStrRev()* - позволяет обнаружить в строковой переменной последовательность символов и вернуть ее позицию. Если последовательность не обнаружена, то возвращается 0;
- *Left()*, *Right()*, *Mid()* - возможность взять указанное количество символов из существующей строковой переменной слева, справа или из середины соответственно;
- *Len()* - возможность получить число символов в строке;
- *LCase()* и *UCase()* - перевести строку в нижний и верхний регистры соответственно;
- *LSet()* и *RSet()* - возможность заполнить одну переменную символами другой без изменения ее длины (соответственно слева и справа). Лишние символы обрезаются, на место недостающих подставляются пробелы;
- *Replace()* - возможность заменить в строке одну последовательность символов на другую.

3. Числовые функции VBA:

- *ABS()* - возвращает абсолютное значение числа;
- *Int()* – возвращает ближайшее меньшее целое число;
- *Fix()* - отбрасывает дробную часть числа;

Round() – округляет до указанного количества знаков после запятой;

- *Rnd()* - используется для получения случайных значений чисел;
- *Sgn()* - возвращает информацию о знаке числа (1 – если число положительное, -1 – если число отрицательное, 0 – если число равно нулю).

4. Функции VBA для работы с датой/временем:

- *Date()* - возвращает текущую системную дату. Установить ее можно при помощи одноименного оператора:

Date = #5/12/2006#

- *Time()* - возвращает текущее системное время;
- *Now()* – возвращает дату и время вместе;
- *DateAdd()* - добавляет к дате указанное количество лет, кварталов, месяцев и так далее - вплоть до секунд;
- *DateDiff()* – позволяет получить возможность получить разницу между датами (от лет до секунд);

- *DatePart()* - возвращает указанную часть даты (например, только год, только месяц или только день недели);
- *MonthName()* - возвращает имя месяца словами по его номеру. Возвращаемое значение зависит от региональных настроек. Если они русские, то вернется русское название месяца.

5. Функции VBA для работы с массивами:

- *Array()* - позволяет автоматически создать массив нужного размера и типа и сразу загрузить в него переданные значения:

Dim myArray As Variant
myArray = Array(10,20,30)

- *Filter()* - позволяет на основе одного массива получить другой, отфильтровав в исходном массиве нужные нам элементы;
- *LBound()* - возвращает информацию о нижней границе массива (номере первого имеющегося значения);
- *UBound()* - возвращает информацию о верхней границе массива(номер последнего имеющегося значения).

Литература

1. Слепцова, Л.Д. Программирование на VBA. Самоучитель / Л.Д. Слепцова. – М. : Издательский дом «Вильямс», 2004. – 384 с. : ил.
2. Малышев, С.А. Самоучитель VBA. Как это делается в Word, Excel, Acces. – СПб. : Наука и Техника, 2001. – 496 с. : ил.
3. Манюкевич, А.В. Лабораторный практикум по курсу Информатика. Программирование на языке VBA для Excel / А.В. Манюкевич. – Минск : БНТУ, 3007. – 47 с.
4. Гарбер, Г.З. Основы программирования на Visual Basic в MS Office 2007 / Г.З. Гарбер. – М. : СОЛОН-ПРЕСС, 2008. – 192 с. : ил.
5. Сафронов, И.К. Visual Basic в задачах и примерах / И.К. Сафронов. – СПб. : БХВ-Петербург, 2006. – 400 с.

СОДЕРЖАНИЕ

Учебное издание

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ VBA

Лабораторный практикум
по дисциплине «Информатика»
для студентов специальности 1-43 01 03
«Электроснабжение (по отраслям)»

С о с т а в и т е л ь
ГОРНОСТАЙ Александр Владимирович

Технический редактор *О.В. Песенько*

Подписано в печать 19.12.2014. Формат 60×84 1/16. Бумага офсетная. Ризография.
Усл. печ. л. 3,89. Уч.-изд. л. 3,05. Тираж 100. Заказ 1214.
Издатель и полиграфическое исполнение: Белорусский национальный технический университет.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя
печатных изданий № 1/173 от 12.02.2014. Пр. Независимости, 65. 220013, г. Минск.