Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Горно-Алтайский государственный университет»

Н. Г. Кудрявцев, Д. В. Кудин, М. Ю. Беликова

ПРОГРАММИРОВАНИЕ НА VBA MS Excel

(Учебное пособие)

Горно-Алтайск
РИО Горно-Алтайского госуниверситета
2015

Печатается по решению редакционно-издательского совета
Горно-Алтайского государственного университета

УДК 004.432
ББК 32.973.26–018.1я73
К 88

Кудрявцев Н.Г., Кудин Д. В., Беликова М. Ю.
Программирование на VBA MS Excel: учебное пособие / Н. Г. Кудрявцев, Д. В. Кудин, М. Ю. Беликова – Горно-Алтайск: РИО ГАГУ, 2015. – 116 с.

Рецензенты :

Власов В.Н., к. ф.-м. н., доцент кафедры общей информатики факультета информационных технологий Новосибирского государственного университета.

Кречетова С. Ю., к. ф.-м. н., доцент, заведующий кафедрой математики и информатики Горно-Алтайского государственного университета.

Пособие содержит наряду с описанием основных конструкций языка программирования VBA MS Excel многочисленные примеры, необходимые для его практического освоения и использования при решении задач автоматизации MS Office. Может быть использовано в качестве основной учебной литературы при изучении дисциплин «Информатика и программирование» (230700 «прикладная информатика»), «Технология программирования и работа на ЭВМ» и «Практикум на ЭВМ (01.03.01 «Математика», 02.03.01 «Математика и компьютерные науки»).

ОГЛАВЛЕНИЕ

1. Введение

Вряд ли можно сказать, что современный человек получил полное образование, если он не владеет иностранными языками. То же самое происходит и в программировании. Профессиональный программист не может быть ограничен одним алгоритмическим языком. Он должен иметь возможность выбирать язык программирования наиболее "удобный" и максимально удовлетворяющий условиям поставленной задачи. Если нужно написать небольшую быстродействующую программу для микроконтроллера, то выбор, скорее всего, будет остановлен на ассемблере. Если микроконтроллерная программа не столь критична ко времени, то ее удобнее всего написать на языке С. Если нужно выполнить разработку какой-то достаточно сложной многомодульной системы для ПЭВМ, то обычно пользуются объектно-ориентированными языками программирования, такими как С++ или С#. Если программируются Интернет приложения, то используют PHP, Perl или Java. Если необходимо обрабатывать и анализировать большие объемы научных данных, то современная тенденция подсказывает использование языка Python. В нашем же случае, при необходимости автоматизировать обработку данных в MS Excel или MS Word, наиболее удобным является язык VBA.

Электронные таблицы MS Excel разрабатывались для упрощения обработки больших объемов числовых данных, для выполнения рутинных операций и наглядного представления результатов вычислений. Даже неподготовленный пользователь может, пользуясь простыми формулами, выполнять бухгалтерские, статистические операции.

Используя особенности продуктов MS Office, при помощи VBA, можно автоматизировать решение практически любой задачи, связанной с этим пакетом программ. Вот только небольшой перечень таких задач:

- оформление результатов наблюдений в таблицах и обработка этих результатов;
- формирование графиков;
- оформление бухгалтерской информации;
- свод различных отчетов, заполняемых в разных местах;
- выделение нужных информационных фрагментов.

VBA позволяет максимально использовать готовые интерфейсные возможности Windows с максимально доступным интерпретатором. Идеология этого языка заключается в том, что основными при реализации процедур автоматизации является не программа, а документ, в котором хранится программа, хотя программы могут храниться и отдельно от документа.

Язык VBA является стандартом де-факто и используется не только в программных продуктах компании Microsoft, но и в свободном программном обеспечении. Например разработка поддержки VBA в пакете офисных программ OpenOffice (LibreOffice) ведется с 2006 года. В последней на момент написания пособия версии LibreOffice 4.3 в ключить поддержку VBA макросов можно в Сервис – Параметры – Загрузка/Сохранение – Свойства VBA.

2. Написание первой программы

VBA позволяет самообучаться при включении автоматической записи макроса.

Сервис/Макрос/Начать запись (для Office 2003)
Разработчик/Макросы (для Office 2007)

Для программирования макросов переходят в окно редактора *VBE*. Проще всего это сделать нажав сочетание клавиш *Alt-F11*

Подавляющее большинство процедур и функций должны быть размещены в *Модулях, Классах, Формах*. Подпрограмма задается ключевым словом *Sub* от слова *Subrouting*

Подпрограмма или процедура:

```
sub имя(параметры, ...)
 ......
end sub
```

Функция:

```
function имя(параметры, ...)  as тип
 имя = out
end function
```

Для того, чтобы функция возвращала искомое значение, в теле функции необходимо записать оператор присваивания переменной, совпадающей с именем функции это искомое значение

вызов процедуры:

```
 call имя_процедуры
```

пример вызова функции с передачей параметров

```
 c=имя_функции(парам_1, парам_2)
```

Если процедура или функция находится в другом модуле, то для их вызова необходимо сначала записать имя модуля, а затем через точку выбрать имя функции или процедуры.

```
Call Module2.proba2
```

Функцию также можно вызвать, поместив ее в ячейку рабочего листа и записав:

```
= имя_функции(A1;B5)
```

Для данного примера в ячейке, в которой размещена приведенная выше запись, появится результат, возвращаемый функцией, а параметры, передаваемые в функцию, будут взяты из ячеек *A1* и *B5*

Комментарии выполняются при помощи апострофа. *Модули, классы, формы* хранятся в документе *xls* или *xla* (файле надстройки).

Процедура от функции отличается тем, что функция возвращает значение, а процедура не возвращает значение

3. Переменные и константы

3.1. Объявление переменных

Переменная – это имя, которое программист дает области компьютерной памяти, используемой для хранения данных определенно размера (типа).

Самым простым способом создания переменной является использование ее в операторе VBA. При этом VBA создает переменную и резервирует память для переменной. Сохранение значения данных в переменной называется присваиванием переменной. Делается это с помощью оператора присваивания (=).

Например: *Sum=120*

Этот оператор сохраняет числовое значение 120 в ячейке памяти, заданной переменной *Sum*. Такое объявление переменной называется *"неявным объявлением переменной"*. Все переменные, которые *VBA* создает подобным образом, имеют тип данных *Variant*.

Для явного объявления переменной используется оператор *Dim (Dimension)*. Ранее в языке *Q-BASIC* этот оператор использовался для объявления размерности массива, отсюда и название – *Dimension* (размерность).

Синтаксис:

```
Dim name1 [, name2…]
```

Все переменные, созданные подобным образом, являются переменными типа *Variant*. Кроме типа *Variant* существуют типизированные переменные (переменные, для которых объявляют тип). Типизированные переменные ускоряют выполнение кода программы. Использование типизированных переменных может значительно сократить объем памяти, требуемой для выполнения программы. Программный код с типизированными переменными легче чи-

тается и понимается. В программах, использующих типизированные переменные, легче обнаружить некоторые ошибки программистов;

Существует функция *VarType(Имя)*, которая возвращает число, соответствующее типу переменной. Ниже в таблице приведены используемые типы переменных, размер выделяемой под них памяти и числа, возвращаемые функцией *VarType*, соответствующие каждому типу *переменной*

Код VarType	Тип переменной	Описание типа
0	*пустая*	пустая инициализированная переменная
1	*NULL*	ошибочное значение
2	*Integer*	целое число, 2 байта
3	*Long*	целое число, 4 байта
4	*Single*	число с плавающей точкой, 4 байта
5	*Double*	число с плавающей точкой, 8 байт
6	*Currency*	чилос с фиксированной точкой, 8 байт, 4 знака после точки
7	*Date*	дата – время, 8 байт, дробное число – целая часть количество дней с 31.12.1899, дробная часть – часы, минуты, секунды * 24
8	*String*	строка длиной от 0 до 65535
9	*Object*	объект
11	*Boolean*	булев тип 1 или 0
17	*Byte*	целое число, 1 байт

Пример использования функции *VarType()*

```
Sub TestVar()
   Stemp = "Hello"
   MsgBox VarType(Stemp)
   Stemp = 4
   MsgBox VarType(Stemp)
 End sub
```

Когда VBA создает новую переменную, эта переменная инициализируется (переменной присваивается некоторое значение «по умолчанию»):

```
строки инициализируются пустыми строками;
числа – значением 0;
переменные типа Boolean – False;
даты – 30 декабря 1899
```

Создаваемые строковые переменные по умолчанию являются строками переменной длины, т.е. они изменяют свою длину, в зависимости от длины стро-

ки, сохраняемой переменной. Но иногда может понадобиться использовать строку фиксированной длины. Такие строки всегда имеют одну и ту же длину. Они полезны, если необходимо обеспечить, чтобы текст, сохраненный в строковой переменной, всегда содержал одно и то же число символов.

Синтаксис:

```
Dim varname As String * N
dim имя1 as тип1, имя2 as тип2
```

3.2. Преобразование типов

В VBA существует *явное и неявное* преобразование типов. Неявное преобразование типов является наиболее удобным и часто употребляемым. Например, если переменная *a* объявлена как целочисленная:

```
Dim a as Integer
```

то при выполнении присваивания этой переменной результата, возвращаемого функцией *InputBox*, строка, например *«52»*, преобразуется в целое число *52*. Если такое преобразование невозможно, то выдается сообщение об ошибке.

```
a=InputBox("","" ,"5")
```

Если же переменная никак не объявлена (имеет тип *Variant*), то преобразования типов не происходит, тип самой переменной преобразуется в тип *String*

У студентов, которые не привыкли объявлять переменные, часто встречается распространенная ошибка:

```
Sub Proba1()
 a=Inputbox("","","5")
 b=Inputbox("","","6")
 c=a+b
 MsgBox c
End sub
```

В этом случае в окне вместо ожидаемого результата *11* появляется результат конкатенации (склеивания) строк: *56*

Неявное преобразование типов происходит также при конкатенации строк с использованием оператора &. Об этом более подробно будет сказано ниже.

Существуют также функции явного преобразования типов, которые используются для преобразования строковых переменных в другие типы данных. В качестве примера можно привести функции:

```
CInt(), CDbl, CDate ....
Dim aDateStr as string, aTimeStr as String
Dim aDate as Date, aTime as Date
aDateStr="2/13/2014"
aTimeStr="5:11:23 AM"
```

```
aDate=CDate(aDateStr)
aTime=CDate(aTimeStr)
```

3.3. Область действия переменных

Любая переменная имеет свою *область действия.*

Термин *область действия* относится к области процедуры или модуля VBA, где данная переменная, процедура или другой идентификатор, являются доступными.

Переменные, процедуры и идентификаторы, которые доступны только в процедуре, имеют область действия процедурного уровня, а те, которые доступны для всех процедур в модуле, имеют область действия модульного уровня.

Область в начале модуля перед любыми объявлениями процедур называют областью объявлений модуля, поэтому туда следует помещать объявления переменных модульного уровня и другие директивы, касающиеся всего модуля.

После выполнения процедуры переменная становится невостребованной, поэтому Excel освобождает соответствующую область памяти. Если требуется сохранить значение переменной, объявите ее как *Static.*

Чтобы сделать переменную доступной во всех процедурах всех модулей VBA в проекте, необходимо объявить переменную на уровне модуля с помощью ключевого слова *Public* перед первой процедурой модуля, например, так:

```
Public CurrentRate as Long
```

Модификатор *Public* должен вводиться в стандартном модуле *VBA*, а не в коде модуля листа или формы.

Если переменная должна быть доступна во всех функциях или процедурах класса или формы, то ее необходимо объявлять перед всеми функциями и процедурами, используя ключевое слово *Dim.*

Константа – именованное значение или строка, которая не меняется при выполнении программы. Константы объявляются с помощью оператора *Const*.

Переменная, определенная как *Date,* является дробным числом, занимает 8 байт памяти и может содержать даты в диапазоне от *1 января 100 года до 31 декабря 9999 года.* Целая часть – количество дней с *31.12.1899,* дробная часть, умноженная на 24 – часы, минуты, секунды текущего времени. В VBA дата и время определяются как значения, заключенные между знаками #.

Если есть желание всегда использовать явное объявление переменных, то необходимо в начале каждого модуля записать команду *Option Explicit*. При описании данной директивы *VBA* требует объявления всех переменных перед их использованием.

Чтобы данная команда автоматически включалась в новый модуль необходимо поставить птичку в следующих настройках

Tools/Options/Editor/Require Variable declaration

Программируя в VBA, можно объявлять структуры или пользовательские типы данных при помощи ключевого слова *Type*. Структура должна быть объявлена в самом начале модуля.

```
Type MyType
   name as String
   age as Integer
End Type
```

Затем в любой функции или процедуре объявляется переменная типа новой структуры и выполняются какие-то действия с элементами структуры

```
Dim a1 as MyType
MyType.name = "Vasya"
MyType.age = 10
```

4. Окно сообщений

Для реализации простейшего пользовательского интерфейса используют окно сообщений *MsgBox*. Ниже будет показано, как данное окно можно использовать и для ввода в программу информации о сделанном пользователем выборе.

Рассмотрим подробнее параметры использования *MsgBox*. Отметим, что в квадратных скобках записаны необязательные параметры.

```
[<p>=] MsgBox (<сообщение> [,<кнопки и значения>]
[,<заголовок окна>] [,<справка, раздел>])
```

p – переменная, которой присваивается код возврата.

Обратите внимание, что круглые скобки используются для описания параметров в том случае, если код возврата присваивается переменной, в остальных случаях аргументы могут быть описаны через пробел.

```
Сообщение              – текст в окне
Кнопки и значок   – vbYesNoCancel + vbInformation
                    + vbDefaultButton3, или3 + 54 + 512
Справка            – имя файла справки
```

```
MsgBox "Привет!", vbYesNoCancel + vbInformation +
          vbDefaultButton3, "Мое первое окно"
MsgBox "Первая программа", , "Окно сообщения"
```

5. Окно ввода InputBox

Функция, реализующая окно ввода, обычно используется для ввода информации и при нажатии на кнопку ОК возвращает значение переменной строкового типа, которая может быть преобразована к нужному типу либо явным, либо неявным образом.

```
[<p>=] InputBox (<сообщение> [,<заголовок>]
   [,<строка по умолчанию>] [,X][,Y] [,<справка>])
```

X,Y – координаты левого верхнего угла в единицах, которые называются *twips (1440 twips - 1 дюйм, 567 twips - 1 см)*

В Таблице 5.1 приведены некоторые параметры интерактивных окон ввода\вывода

Таблица 5.1

Кнопки\значки	Код	Значение
vbOKOnly	*0*	Только ОК
vbOKCancel	*1*	ОК, Отмена
vbAbortRetryIgnore	*2*	Стоп, Повтор, Пропустить
vbYesNoCancel	*3*	Да, Нет, Отмена
vbYesNo	*4*	Да, Нет
vbCritical	*16*	Критическое
vbQuestion	*32*	Вопрос
vbExclamation	*48*	Оповещение
vbInformation	*64*	Информация

В Таблице 5.2 содержатся проверочные служебные слова.

Таблица 5.2. Проверочные значения

Проверочные значения
vbOK
vbCancel
vbAbort
vbRetry
vbIgnore
vbYes
vbNo

6. Операции

6.1. Арифметические

Совершенно очевидно, что процесс обработки и преобразования числовой информации не может быть реализован без арифметических и логических операций, которые представлены ниже.

Таблица 6.1

Операция	Описание
^	возведение в степень
–	отрицание
*	умножение
/	деление
\	целочисленное деление
mod	остаток от деления (операция по модулю)
+, –	арифметические операции
&	объединение строк с преобразованием типов
+	объединение строк (без преобразования типов)
_	перенос строки

Примеры использования операций

```
Dim s as string
s="proba " + 3          'ошибка
s="proba " + "3"        'корректно
s="proba " & 3          'корректно

17 / 5       = 3.4
17 \ 5       = 3
17 mod 5 = 2
```

6.2. Сравнения

Таблица 6.2

Операция	Описание
=	Равно
<>	Не равно
<	Меньше
>	Больше
<=	Меньше или равно
>=	Больше или равно

6.3. Полезные встроенные функции

```
Abs()     модуль числа
Round(a,b)    округляет число a до b десятичных знаков
Int()     округление до целого числа
Fix()     округление до целого числа
Int(99.8) '99
Fix(99.8) '99
Int(-99.8)    '-100
Fix(-99.8)    '-99
Int(-99.2)    '-100
Fix(-99.2)    '-99

Sign()    возвращает знак числа
Sqr()     квадратный корень
log()     десятичный логарифм
```

7. Массивы

Массив - это набор (множество) однотипных данных. Обычно в компьютерной памяти элементы массива расположены рядом для удобства и скорости доступа.

Ниже показан пример объявления целочисленного одномерного массива Mas1 размерности 6 и целочисленного двумерного массива Mas2 размерности 5x6. Нумерация, по умолчанию, начинается с нулевого элемента, цифра в скобках указывает на значение верхнего граничного элемента. Если необходимо изменить значение номера нижнего элемента по умолчанию, то использую директиву *Option Base 1 (в этом случае элементы будут нумероваться с 1)*:

```
Dim Mas1(5) As Integer
Dim Mas2(4,5) As Integer
```

Если необходимо определенно указать с какого и по какой элементы массива будут использоваться, то допускается следующая запись:

```
Dim Mas3(2 to 12) As Integer
```

При необходимости изменить размерность массива можно воспользоваться оператором *ReDim.*

```
ReDim MyArr(10)
```

При таком изменении размерности массив очищается (практически под него выделяется новое место в памяти без переноса данных). Если необходимо, чтобы в массиве сохранились записанные ранее элементы, то используют ключевое слово *Preserve.* Однако в этом случае можно изменить только верхнюю границу массива.

```
ReDim Preserve MyArr(100)
```

Если заранее не известен размер массива, то используют так называемые динамические массивы (объявляют массив пустой размерности):

```
Dim MyArr2() as Integer
```

После того, как размерность становится известной, используем оператор *ReDim*. Необходимо отметить, что с оператором *ReDim* можно использовать только целочисленные переменные в качестве параметров объявления размерности массива.

```
Dim MyArr2()
a=3
b=4
ReDim MyArr2(a,b)
```

В VBA существуют функции, которые возвращают номера соответственно минимального и максимального элементов массива:

`LBound()` и `UBound()` соответственно

При необходимости можно проинициализировать (одним оператором задать) элементы массива

```
Dim MyArr1 as Variant
MyArr1 =Array(1,2,3,4,5)
Dim MyArr() as Variant
MyArr = [{1,2,3,4,5}]
```

В данном случае инициализируется одномерный массив из пяти элементов. Надо отметить, что нумерация элементов при такой инициализации начинается с единицы

Если выполнить команду `ReDim Preserve MyArr(4)`, то нумерация, по умолчанию, начнется с нуля.

В следующем примере показана инициализация двумерного массива

```
Dim MyArr as Variant
MyArr = [{1,2,3;4,5,6;7,8,9}]
```

При использовании функций или процедур для работы с внешними массивами можно использовать массивы, объявленные как глобальные переменные, однако, лучше передавать их в процедуру или функцию в качестве параметра по ссылке *(ByRef)*.

При передаче массива в функцию или процедуру по ссылке объявление параметра выглядит следующим образом:

```
sub MasProc(ByRef a()) as Integer)
```

8. Циклы

Обычно циклы используют для многократного выполнения каких то действий. Количество повторений может быть задано заранее, либо может определяться каким-то одним, или несколькими условиями.

8.1. Цикл For ... Next

Используется для выполнения заданного количества повторений. При этом, параметр цикла (цикловая переменная) может увеличиваться или уменьшаться на определенную величину, определяемую параметром *Step (шаг)*. По умолчанию шаг равен 1.

```
For i = 1 To 5
  MsgBox i
Next i

For n=3 to 12 Step 3
   MsgBox n
Next n

For n=10 to 1 Step -1
     MsgBox n
Next n
```

8.2. Цикл For Each ... Next

Используется для работы с коллекциями и диапазонами. Данный цикл «проходит» по каждому элементу заданной коллекции или диапазона.

```
Sub ShowName()
   Dim  oWSheet as WorkSheet
   For Each oWSheet in WorkSeets
      MsgBox oWSheet.Name
   Next oWSheet
End Sub
```

8.3. Цикл While ... Wend

Цикл выполняется до тех пор, пока условие выполняется (истинно). До тех пор пока ... Условие данного цикла проверяется перед входом в цикл (перед выполнением повторяющихся действий).

```
Sub test_while()
  x=0
  While  x<10
    x=x+1
```

```
    MsgBox x
  Wend
End Sub
```

8.4. Цикл Do ... Loop

Данный цикл имеет несколько модификаций: с предусловием и с постусловием.

Конструкция цикла	Описание
Do While ... Loop	Цикл с предусловием выполняется до тех пор, пока условие «истинно»
Do Until ... Loop	Цикл с предусловием выполняется до тех пор, пока условие «ложно»
Do ... Loop While	Цикл с постусловием выполняется до тех пор, пока условие «истинно»
Do ... Loop Until	Цикл с постусловием выполняется до тех пор, пока условие «ложно»

Цикл с постусловием выполняется до тех пор, пока условие «ложно».

```
Sub doTest()
  x=80
  Do Until x=100
    x=x+1
  Loop
  MsgBox x
End Sub
```

Циклы, процедуры и функции можно прерывать. Прерывание циклов может понадобиться, если ведется поиск какого-то элемента в массиве и этот элемент найден. Если циклы вложенные, то выход происходит только из одного цикла.

```
For ...  Next For Each ... Next    Exit For
Do ... Loop                        Exit Do
```

Конструкции *Exit sub* и *Exit Function* позволяют прервать выполнение процедуры и функции при выполнении или не выполнении каких-то условий

9. Управление потоком выполнения программы

При необходимости реализовать выбор между различными ветвями (потоками) выполнения программы, используют условные операторы.

9.1. Оператор If ... Else ... End If

```
If  InX > 0 Then
  ...
```

```
Else
  ...
End If
```

Условные операторы могут иметь вложенную структуру. Используя вложенные конструкции, надо помнить, что каждый *If (открывающий оператор)* должен иметь свой *End if* (закрывающий оператор). Для простоты соблюдения этого правила, код программы форматируют таким образом, чтобы все операторы, находящиеся между «открывающим» и «закрывающим» операторами были сдвинуты вправо, либо на символ табуляции, либо на несколько пробелов.

```
If  InX = 0 Then
  If  InY < 5 Then
    …..
  Else
    ….
  End If
  ....
Else
  ….
End If
```

Если после *Then* должен следовать единственный *оператор*, то данная конструкция может быть записана в одну строку без закрывающего оператора *End If:*

```
If   InX  >  0 Then  ОПЕРАТОР
```

```
Условные операторы    =,  <, >, <>, <=, >=,
Сложные условия:      And, OR
```

В некоторых случаях возникает необходимость реализовать множественный выбор (выбор между несколькими значениями одного условия). В этом случае используют конструкцию *Select Case*

9.2. Оператор Select Case

```
Select Case i
  Case 1
    …..
  Case 2
    ….
End Select

x=23
Select Case(x)
  Case 1
```

```
    MsgBox "x=1"
    ...
  Case 23
    MsgBox "x=23"
End Select


Select Case(Grade)
  Case 1
    ....
  Case 2,3
    ....
  Case 4 to 6
    ....
  Case Is > 8
    ....
  Case Else
    ...
End Select
```

Ниже приведен пример проверки и обработки ячеек в заданном регионе. Если значение записанного в ячейку числа меньше 10, то число становится зеленым

```
Sub wwx()
Workbooks("exp1.xls").worksheets("Лист2").Activate
 For Each c In Range("a1:c15")
    If c.Value < 10 Then c.Font.ColorIndex = 4
 Next c
End Sub
```

10. Работа со строками

Большое количество задач, возникающих при решении проблемы автоматизации как экономико-бухгалтерской, так и научно исследовательской деятельности, связано с обработкой текстов, а точнее, с работой со строковыми переменными.

10.1. Функции преобразования типов

Ниже приведены несколько функций, преобразующих строки в числа и числа в строки

```
Val("56")+Val("1.1")           ' 57,1
CInt("56")+CDbl("1,1")         ' 57,1
Str(764)                   ' " 764"
CStr(764)                  ' "764"
```

Необходимо обратить внимание на некоторые различия между показанными выше функциями. Например функции *Val()* и *CDbl()* по разному обрабатывают разделители между целой и дробной частью дробных чисел. Одна функция требует точку, другая - запятую. Функция *Str()* добавляет пробел в начало преобразованной строки, при этом, функция *CStr()* преобразует число в строку «без изменений»

```
len(str(764))              ' 4
len(Cstr(764))             ' 3
len("764")                 ' 3
```

10.2. Форматирование строк

Для более удобного представления числовых данных при преобразовании в строковую переменную используют функцию *Format()*, которая производит форматирование выводимой информации по заданному шаблону.

```
MsgBox Format(1234567.89,"#,###.#")    '1 234 567.9
MsgBox Format(1234567.89,"Currency")   '1 234 567.89р
```

10.3. Разбиение строк

Одной из наиболее часто встречающихся задач при работе со строками (например, при чтении строк из файла) является задача разбиения строки на подстроки по заданному разделителю (разбиение строки на слова по пробелу). Достаточно просто эта задача решается при помощи функции *Split()*. Для ее использования необходимо объявить строковый динамический массив, в который будут записаны подстроки, полученные в результате работы данной функции.

```
Dim MyArray() as String
StrBuf  = "1232 12313 243 sdfs"
MyArray = split(StrBuf," ") 'разбор
sz = UBound(MyArray) 'количество найденных фрагментов
```

10.4. Поиск подстроки

Также достаточно часто для поиска позиции вхождения подстроки в строку используется функция *InStr()*. Если ничего не найдено, то данная функция возвращает 0, если найдена искомая подстрока, то будет возвращена позиция первой найденной подстроки. Для поиска повторных вхождений подстроки в строку используют первый параметр функции *InStr()*, который указывает с какой позиции начинать поиск. По умолчанию этот параметр равен 1.

```
p1                  - начальная позиция поиска
p2                  - найденная позиция
IStr                - искомая строка
StrBuf              - строка источник
```

```
p2 = InStr(p1, StrBuf, IStr)
```

10.5. Копирование и замена подстроки

Для копирования (сохранения) части строки в другую строку используют, в зависимости от места вхождения копируемой строки, три функции: *Left()*, *Right()*, *Mid()*. Ниже приведены примеры выделения (сохранения) подстроки *Shablon* из строки *StrInput*, если *Shablon* находится в начале строки, в конце строки и в середине, соответственно.

```
StrOut = Left(StrInput, Len(Shablon))
StrOut = Right(StrInput, Len(Shablon))
Begin = InStr(1,StrInput,Shablon)
StrOut = Mid(StrInput, Begin, Len(Shablon))
```

Также существуют функции, которые позволяют удалять пробелы в начале строки, в конце строки и в начале и конце строки, соответственно: *LTrim()*, *RTrim()*, *Trim()*

Функция *Replace(StrBuf,str1,str2)* позволяет заменять в строке *StrBuf* подстроку *str1* на строку *str2*

10.6. Сравнение строк при помощи оператора Like

Иногда требуется производить сравнение строк не полностью, а в соответствии с шаблоном. В этом случае полезно использовать оператор *Like*

```
return = string Like pattern
Элементы для pattern      ?   - единичный символ
          *   - 0 или любое кол-во символов
          #   - любая цифра
          [ ]  - набор символов
          [! ]- любой символ не из набора
```

Таблица 10.1.

Строка	Параметр Like	Возвращаемое значение
"aBBBa"	*"a*a"*	*True*
"F"	*"[A-Z]"*	*True*
"F"	*"[!A-Z]"*	*False*
"a2b"	*"a#a"*	*True*
"aM5b"	*"a[L-P]#[!c-e]"*	*True*
"BAT123khg"	*"B?T*"*	*True*
"CAT123khg"	*" B?T*"*	*False*

10.7. Другие полезные функции для работы со строками

Ниже приведем еще несколько функций работы со строками. Функция *Len()* возвращает длину строки. Функция *Asc()* позволяет получить код символа, а функция *Chr()* - символ по коду

```
Len()     длина строки
Asc()     код символа
          Asc ("a") = 97
          Asc ("A")= 65
Chr()     Возвращает строку, состоящую из символа,
            код которого задан в качестве параметра
```

11. Работа с датой и временем

Чтобы получить значение текущей даты и текущего времени или даты и впремени по отдельности, используют следующие функции:

```
Now()
Date()
Time()
```

Существуют также функции, позволяющие выделить из общей даты отдельные ее значения, такие как день, часть, минуту и т.п.

```
Dim a1 as date
   a1 = Now()
   dm = Minute(a1)
   dh = Hour(a1)
   dd = Day(a1)
   mm = Month(a1)
   dy = Year(a1)
```

Можно также добавлять к дате определенные временные интервалы *DateAdd* или находить разницу между двумя датами *DateDiff*

```
dataizm = Now() 'присваивание строковой переменной
                'dataizm значения текущей даты
cc = DateAdd("m", a, DateValue(ddX))
                  'преобразование строковой
                  'переменной ddX к формату
                  'хранение даты и прибавление
                  'к ней количество месяцев равное a
ddX = Format(cc, "mmmm yyyy")
                'преобразование даты к
                'определенному формату
MsgBox Format(Now(),"hh:mm:ss Am/Pm")
                 'mm/dd/yy    - 01/03/06
```

```
                'dd-mmm-yyyy 01-Mar-2012

Print "Интервал в годах = " & _
DateDiff("yyyy", DateStart, DateFinish)
Print "Интервал в месяцах= " & _
DateDiff("m", DateStart, DateFinish)
Print "Интервал в днях = "& _
DateDiff("d", DateStart, DateFinish)
Print "Интервал в часах = "& _
DateDiff("h", DateStart, DateFinish)
Print "Интервал в минутах = "& _
DateDiff("n", DateStart, DateFinish)
Print "Интервал в секундах= "& _
DateDiff("s", DateStart, DateFinish)
```

12. Взаимодействие MS Exel с другими приложениями

В большинстве случаев для долговременного хранения обрабатываемой в приложениях *MS Office* информации наряду с базами данных могут быть использованы как специальные *xls, doc (xlsx, xlsm, docx, docm)*, так и обыкновенные текстовые файлы. В данном разделе рассмотрим как работать с такими объектами: открывать, закрывать, сохранять информацию, получать доступ к уже хранящейся информации.

12.1. Работа с файлами MS Excel

Сначала рассмотрим как работать с файлами *MS Excel*. Базовым понятием в этом вопросе является рабочая книга *Workbook*, точнее коллекция рабочих книг *Workbooks*. Ниже, когда речь пойдет об объектной модели *MS Office*, мы подробнее рассмотрим этот вопрос.

На нескольких примерах открывания файлов *xls (xlsx, xlsm)* мы хотим показать обязательные и необязательные параметры, которые при этом используются, и различные способы передачи этих параметров. Подробнее о методах объекта *Workbooks* можно прочитать в справочной литературе.

```
Workbooks.Open("C:\MyFile.xls")
                  'имя файла - обязательный параметр,
                   'пароль - необязательный параметр
Workbooks.Open "C:\MyFile.xls", , True, "orange"
                  'имя только чтение  пароль
Workbooks.Open FielName:="C:\MyFile.xls",
                   'ReadOnly:=True, Password:="orange"
```

В первых примерах открывания файла показана передача параметров *«по месту»*, в последнем примере - передача параметров «*по имени»* (такой способ является в некоторых случаях более удобным).

Далее покажем как сохранять файл:

```
Workbooks("book1").Save -        нет обязательных параметров
Workbooks("book1").SaveAs("NewFile.xls", , "orange")
```

– обязательным является только первый параметр. Второй параметр - формат файла, необязательный, его пропускаем, третий параметр - пароль, необязательный, его задаем – «orange».

Обычно для того, чтобы избежать ошибок при выполнении операции, перед открыванием файла проверяют его наличие по заданному пути. Для проверки используют функцию: `DIR(pathx+fName)`. Эта функция возвращает строку - имя файла, если файл с именем `fName` находится по пути `pathx`. Если искомого файла по указанному пути нет, то возвращается пустая строка

Ниже приведен пример проверки наличия и открытия нескольких файлов

```
For i = 1 To all
 StrFile = Dir(path_x + mybook(i))
 If StrFile = "" Then
   MsgBox "Не могу найти файл:" & path_x & mybook(i)
     Exit Function
  End If
Next i
For i = 1 To all
  Workbooks.Open FileName:=path_x + mybook(i)
Next i
```

Ниже приведен пример закрытия файла с сохранением. Если параметр `SaveChanges` примет значение `FALSE`, то файл закроется без сохранения

```
For i = 1 To all
  Workbooks(mybook(i)).Close SaveChanges:=True
Next I
```

Ниже приведен пример работы с файлом шаблоном. Открываем заранее подготовленный файл и сохраняем его с другим именем. Далее мы можем изменять активную рабочую книгу, не внося изменения в шаблон.

```
Workbooks.Open FileName:=path_x + bookRazrabTempl
Workbooks(bookRazrabTempl).Activate
ActiveWorkbook.SaveAs FileName:=path_x + BookRazrab
```

Выбрать имя открываемого или сохраняемого файла и путь к нему можно и в интерактивном режиме, используя методы объекта `Application`

```
StrFileName = Application.GetOpenFilename
fName = Application.GetSaveAsFilename
```

В данном случае для получения имени файла используется диалог. Диалог прерывается, если пользователь выберет какое-то имя, либо нажмет кнопку от-

мена. Если ничего не введено и нажата кнопка ОК, то диалог продолжается. Если выбрано имя, то возвращается путь и имя, если нажата кнопка отмена, то возвращается значение *False*. Приведнный ниже пример демонстрирует, как заставить пользователя выбрать хоть какое-то имя.

```
Sub ccc()
  Do
   fName = Application.GetOpenFilename
  Loop Until fName <> False
  MsgBox "Opening " & fName
  Set myBook = Workbooks.Open(FileName:=fName)
  myBook.Close SaveChanges:=False
End Sub
```

Ниже показан пример создания и сохранения нового файла

```
Sub CreateAndSave()
  Dim newBook as Workbook
  Set newBook = Workbooks.Add
  Do
    fName = Application.GetSaveAsFilename
  Loop Until fName <> False
  newBook.SaveAs FileName:=fName
End Sub
```

После того как файл открыт, начинается фаза обработки информации. Для данного типа файлов *(xls, xlsx)* пользователь часто сам непосредственно обращается к ячейкам электронных таблиц, реже происходит работа через пользовательские формы, о которых речь пойдет ниже. Программы, автоматизирующие обработку информации, в подавляющем большинстве случаев должны «уметь» записывать и читать информацию из определенных одиночных ячеек, либо из «групп» ячеек рабочего листа (объект *Worksheet* или элемент коллекции *Worksheets*)

Для доступа к ячейке рабочего листа используют либо элемент коллекции *Cells()*, либо объект *Range*. Ниже приведены примеры чтения/записи в ячейки рабочего листа с использованием различных методов доступа. Сразу обратим внимание на то, что перед тем, как осуществлять доступ к ячейкам, необходимо активировать нужный рабочий лист (*Worksheet)* нужной рабочей книги (*Workbook)*

```
Workbooks("MyFile.xls").Worksheets("MyWorksheet")
                                              .Activate
Cells(1,1).Value = 5        ' число 5 записывается в
                             ячейку A1
Range("A3:B5").Value=10     ' число 10 записываетя в 6
                            ячеек первого и второго'
```

```
                                столбцов в строки с
                                третьей по пятую.
a=Cells(1,3).Value              'содержимое ячейки A3
                                 записывается в перемнную a
```

В данном примере при обращении к коллекции ячеек *Cells* в качестве первого параметра передается номер строки (начиная с первого), вторым параметром - номер столбца (начиная с первого)

12.2. Работа с файлами MS Word

Работать из *MS Excel* с файлом текстового процессора *MS Word* несколько сложнее, чем с родным файлом MS Excel или текстовым файлом. Ниже на простом примере покажем как можно решить данную задачу.

Сначала надо создать объект *Word.Application*, сделать его видимым, затем вызвать метод *Open* коллекции *Documents*, относящейся к данному объекту. Ссылку на объект, которую возвращает метод *Open* желательно присвоить объектной переменной, чтобы через эту переменную обращаться в дальнейшем с открытым документом. Закрыть документ можно, используя метод *Close*, при этом указав одно из значений необязательного параметра *(wdSaveChanges/wdDoNotSaveChanges).* Также новый документ можно сохранить, используя метод *SaveAs*.

Запись и чтение информации при работе с *MS Word* осуществляется многими различными способами, которые зависят от объектов уровня приложения или документа, с которыми Вы хотите работать. Если вы хотите что-то записать в самое начало документа и у вас нет никаких других объектов, то можно использовать метод *TypeText* объекта *Word.Application. Selection*

```
MyApp.Selection.TypeText="Hello"
```

Этот же метод позволит вам добавить текст в самое начало документа, не меняя его остальное содержимое. Вы можете воспользоваться возможностью изменить значение свойства *Text* объекта ***Range*** уровня *документа*

```
MyDoc.Range.Text= "Hello"
```

Но после выполнения данной команды весь документ будет содержать только одно слово *"Hello"* Отметим, что свойство *Text* есть у объекта *Word,* у объекта ***Range***, который по иерархии подчинен таким коллекциям как *Paragraphs* или таким объектам как *Tables.Item.Cell*

Получить доступ к информации хранящейся в документе на чтение, можно также используя свойство ***Text*** различных объектов документа.

Покажем простейший пример создания документа и записи в него одной фразы.

```
Sub mMyMacros()
  Dim mWord
  Set mWord = CreateObject("Word.Application")
  mWord.Visible = True
  mWord.Documents.Add
  mWord.Selection.TypeText ("Здравствуй мир!")
End Sub
```

Ниже приведен пример, в котором показано как открыть нужный документ, вписать текст в начало документа, прочитать текст, относящийся к объектам различного уровня и дописать текст в конец документа.

```
Sub WordX1()
 Dim myA As Word.Application
 Dim myD As Document
 Set myA = CreateObject("Word.Application")
 myA.Visible = True
 pathx = ThisWorkbook.Path + "\"
 Set myD = myA.Documents.Open(pathx +
                                  "lecture2.docx")
 myA.Selection.TypeText ("Hello WORLD")
 ss = myD.Words(1).Text
 MsgBox "Word: " & vbLf & vbLf & ss
 ss = myD.Paragraphs(1).Range.Text
 MsgBox "Paragraph: " & vbLf & vbLf & ss
 ss = myD.Range.Text
 MsgBox "Document: " & vbLf & vbLf & ss
 myD.Paragraphs.Add
 myD.Paragraphs.Last.Range.Text = "Конец текста"
 myD.Close SaveChanges:=True
 myA.Quit
End Sub
```

Еще один весьма полезный объект для работы с текстовым процессором *MS Word* - это *объект Bookmark* (закладка). На практике это самый удобный способ навигации по документам, созданным при помощи шаблонов (например, отчетов). Принципиальное отличие его от объектов *Selection и Range* заключается в том, что все выделения и диапазоны теряются при закрытии документа (объекты *Range* вообще существуют только во время работы создавшей их процедуры, а закладки сохраняются вместе с документом. Если документ создан на основе шаблона, то все закладки, которые были определены в этом шаблоне, будут определены и в созданном на его основе документе.

Создать закладку (с помощью меню *Вставка | Закладка*) намного проще, чем считать количество символов для объекта Range от начала документа, абзаца или предложения или выполнять операции *Move()*

(MoveDown(), MoveRight(), MoveNext()) для объекта *Selection*. Кроме того, если вы будете исправлять шаблон (а делать это приходится очень часто), вам, скорее всего, не придется править код для определения места вставки (что потребуется для объектов *Selection и Range*).

Свойств и методов у объекта *Bookmark* существенно меньше, чем у объектов *Selection* и *Range*. Однако, обычно никто и пытается использовать объект *Bookmark* для работы с текстом. Из объекта *Bookmark* (все закладки собраны в коллекцию *Bookmarks*, принадлежащей документу) очень просто получить объект *Selection* (при помощи метода *Select()*) или объект *Range* (при помощи свойства *Range)*, и дальше можно пользоваться уже свойствами и методами этих объектов, например:

```
ThisDocument.Bookmarks("Bookmark1").Select
MsgBox Selection.Text
```

Создавать объекты Bookmark программным способом необязательно, но если есть необходимость, то можно использовать метод *Add()* коллекции *Bookmarks:*

```
ThisDocument.Bookmarks.Add Name:="Proba",
Range:=Selection.Range
```

У этого метода всего лишь два параметра, которые и используются в примере.

Некоторые важные свойства объекта *Bookmark* представлены далее.

Empty — если это свойство возвращает *True*, то закладка указывает на указатель вставки, а не на текст;

Name — имя закладки. Очень удобно, что найти нужную закладку в коллекции Bookmarks можно не только при помощи индекса (номера) закладки, но и по ее имени.

Range — возвращает объект *Range* на месте этой закладки.

Start, End, StoryType — эти свойства аналогичны таким же у объекта *Selection.*

Методов у объекта Bookmark всего три.

Copy() — создает закладку на основе существующей.

Delete() — удаляет закладку.

Select() — выделяет то, на что ссылается закладка.

Ниже приведем пример работы с закладками, заданными в файле шаблона.

```
Sub WordX2()
  Dim myA As Word.Application
```

```
    Dim myD As Document
    Set myA = CreateObject("Word.Application")
    myA.Visible = True
    pathx = ThisWorkbook.Path + "\"
    Set myD = myA.Documents.Open(pathx + "dogovor.docx")
    myD.Bookmarks("bNumber").Range.Text = "23"
    myD.Bookmarks("bCity").Range.Text = "Горно-Алтайск"
    myD.Bookmarks("bDate").Range.Text = "30.11.2014"
    myD.Bookmarks("bOrg").Range.Text = "ГАГУ"
    myD.Bookmarks("bPerson").Range.Text = "Бабин В.Г."
    myD.Bookmarks("bTitle").Range.Text = "Разработка и
    создание"
    myD.Close SaveChanges:=True
    myA.Quit
End Sub
```

Еще одним полезным объектом документа *MS Word* является таблица. С ее помощью можно не только выводить табличную информацию, но и, скрыв границы, получать произвольное размещение на листе различных объектов.

Создание таблицы начинается с того, что в коллекцию *Tables* (она предусмотрена для объектов *Document, Selection и Range)* добавляется новый объект *Table* (в данном примере с тремя строками и четырьмя столбцами):

```
Set Range1 = ThisDocument.Range(Start:=0, End:=0)
Dim Table1 As Table
Set Table1 = ThisDocument.Tables.Add(Range1, 3, 4)
```

Затем можно настроить свойства таблицы, например, воспользовавшись методом *AutoFormat()* (возможности у него те же, что доступны через меню *Таблица* | *Автоформат*):

```
Table1.AutoFormat wdTableFormatGrid5
```

Чаще всего нам нужно ввести какие-либо данные в ячейку таблицы. Ячейку таблицы в *Word* представляет объект *Cell.* Мы можем добраться до нужной ячейки через объекты *Columns* и *Rows, Selection и Range*, однако удобнее всего сделать так:

```
Table1.Cell(1, 1).Range.InsertAfter "10"
Table1.Cell(2, 1).Range.InsertAfter "15"
Table1.Cell(3, 1).AutoSum
```

Мы ввели в первую строку первого столбца значение 10, во вторую строку первого столбца — значение 15, а в третьей строке мы просуммировали значения по всему столбцу. Таблицы *Word* — это, конечно, не *Excel*, но при по-

мощи метода *Formula()* объекта *Cell* в таблицу можно вставлять достаточно сложные вычисляемые значения.

12.3. Работа с базами данных

Для взаимодействия с другими прикладными программами, не относящимися к приложениям *MS Office,* можно воспользоваться базами данных. Ниже, говоря о доступе к базам данных, мы не будем останавливаться на том, что такое базы данных, что такое Системы управления базами данных, что такое Структурированный язык запросов *SQL*, что такое отношения, ключевые поля, связи между отношениями. Всю эту информацию можно более подробно узнать из других источников. Наша задача показать только технологию доступа к базам данных.

12.4. Работа с базами данных при помощи DAO (Data Access Object)

Перед тем как приступить к работе, необходимо перейти в режим редактирования *VBA*. В меню *Tools\References* выбрать библиотеку *Microsoft DAO 3.6 Library.* Также необходимо использовать оператор *Dim* для создания новых объектных переменных для каждого объекта в базе данных. Один объект *Database*, один объект *Workspace*;

```
Dim MyDB As Database
Dim MyWs As Workspace
```

В некоторых случаях существует необходимость создать базу данных из программы. Для этого можно использовать метод *CreateDatabase* объекта *Workspace.* Для метода *CreateDatabase* желательно использовать три аргумента: Название базы данных (путь к файлам базы); тип национальных стандартов (кодировок) в нашем случае это будет *dbLangCyrilic*; версия базы данных (в нашем случае *dbVersion40*). Также можно использовать строку соединения (либо использовать готовое соединение). В этой строке указывается пароль, источник данных, версия *Microsoft Jet.*

```
Set MyWs = DBEngine.Workspaces(0)
Set MyDB = MyWs.CreateDatabase("DBPoba1.mdb",
dbLangCyrillic, dbVersion40)
```

Пример, рассмотренный ниже показывает, как создать базу данных

```
Sub Proba_DB1()
  Dim pathx As String
  Dim MyWs As Workspace
  Dim MyDb As Database
  pathx = ActiveWorkbook.path + "\"
  Set MyWs = DBEngine.Workspaces(0)
  If Dir(pathx + "DBproba1.mdb") <> "" Then
    Kill pathx + "DBproba1.mdb"
```

```
  End If

  Set MyDb = MyWs.CreateDatabase(pathx + "DBpr & _
    ba1.mdb",dbLangCyrillic,dbVersion40)
  MyDb.Close
  MyWs.Close
End Sub
```

Если мы создали базу данных, то необходимо создать таблицу или несколько таблиц для хранения и выборки информации. Существует много способов создания таблиц, также существует много дополнительных параметров, связанных с этой задачей. В нашем случае рассмотрим простейший вариант создания одиночной таблицы с использованием структурированного языка запросов *SQL*. Обратим внимание на то, что при работе с *DAO* мы взаимодействуем с базой данных представленной в виде файла. Для выполнения этой задачи объявляются объектные переменные, связывающие программу с базой данных, открывается база данных. Затем выполняется запрос на языке *SQL*, составленный определенным образом.

Пусть необходимо создать таблицу *Weather* (погода) в уже существующей базе данных (Таблица 12.1)

Таблица 12.1

Наименование поля	Тип данных
Key_weather	*Счетчик*
Data_weather	*Дата\время*
Temperature_weather	*Числовой с плавающей точкой*
Pressure_weather	*Числовой с плавающей точкой*
Humiditi_weather	*Числовой с плавающей точкой*

```
Sub Proba_DB2()

Dim strSQL As String
Dim pathx As String
Dim MyWs As Workspace
Dim MyDb As Database

pathx = ActiveWorkbook.path +"\"
Set MyWs = DBEngine.Workspaces(0)
Set MyDb = MyWs.OpenDatabase(pathx + "DBproba1.mdb")

strSQL = "Create table weather ([key_weather] COUNTER
CONSTRAINT key_weather PRIMARY KEY, [data_weather]
DATETIME,[temperature_weather] DOUBLE, [humidi-
ty_weather] DOUBLE, [pressure_weather] DOUBLE)"
```

```
MyDb.Execute strSQL
MyDb.Close
MyWs.Close
End Sub
```

Теперь, когда таблица создана, можно добавлять в нее информацию. Для этого также используем язык *SQL* и запрос *Insert*. В примере, показанном ниже сначала объявляются уже знакомые нам объектные переменные, затем открывается существующая база данных и выполняется заранее подготовленный запрос на добавление одной строки информации.

```
Sub proba_DB3()
  Dim strSQL As String
  Dim pathx As String
  Dim MyWs As Workspace
  Dim MyDb As Database
  pathx = ActiveWorkbook.path
  Set MyWs = DBEngine.Workspaces(0)
  Set MyDb = MyWs.OpenDatabase(pathx +
  "\DBproba1.mdb")

  strSQL = "Insert into weather (data_weather,
            temperature_weather, humidity_weather,
            pressure_weather)values (#11/14/2013
            11:23:00#,-13.1,77.5,748.2)"

  MyDb.Execute strSQL
  MyDb.Close
  MyWs.Close
End Sub
```

Теперь, когда информация добавлена в базу данных, покажем как выбирать оттуда информацию. Отличие от предыдущих примеров заключается в том, что мы только передавали в базу данных команды или информацию и ничего из базы не получали. В данном же случае мы должны получить из базы результат запроса и заранее не знаем, какого объема будет переданная нам информация. Поэтому при осуществлении выборки информации из базы данных полученная информация хранится в буфере, который называется *Recordset* (набор записей) и уже от пользователя зависит, какой частью информации он воспользуется. Для получения информации из набора *Recordset* можно перемещаться по одной записи вперед и назад (зависит от типа открытого рекордсета), можно обращаться к отдельным полям, можно узнать сколько записей было выбрано из базы данных. Отметим, что на самом деле объект *Recordset* используется не только для выборки информации, но и для редактирования и

добавления информации, однако эти свойства рекордсета в данном пособии рассматриваться не будут. В примере, приведенном ниже выбирается информация о дате и температуре и выводится в рабочий лист рабочей книги *MS Excel*

```
Sub proba_DB4()
  Dim i As Integer
  Dim strSQL As String
  Dim pathx As String
  Dim MyWs As Workspace
  Dim MyDb As Database
  Dim rst As Recordset
  path = ActiveWorkbook.path +"\"
  Set MyDb = MyWs.OpenDatabase(path + "DBproba1.mdb")
  ActiveWorkbook.Worksheets("Лист1").Activate


  strSQL = "Select data_weather, temperature_weather
           from weather"
  Set rst = MyDb.OpenRecordset(strSQL, dbOpenSnapshot)
  i = 1
  Do Until rst.EOF()
    Cells(i, 1).Value = rst!data_weather
    Cells(i, 2).Value = rst!temperature_weather
    rst.MoveNext
    i = i + 1
  Loop
  MyDb.Close
  MyWs.Close

End Sub
```

Ниже приведен пример, собранный из показанных выше фрагментов, показывающий как создается сначала база данных, затем таблица для хранения информации о погоде, затем в эту таблицу добавляется информация о двадцати четырех измерениях температуры, давления и влажности, хранящаяся в соответствующих массивах. Затем эта информация, выбранная из базы данных в соответствии с запросом, помещается в рабочий *Лист1* текущей (активной рабочей книги)

```
Sub proba_DB5()
  Dim i As Integer
  Dim strSQL As String
  Dim pathx As String
  Dim MyWs As Workspace
  Dim MyDb As Database
  Dim rst As Recordset
```

```
Dim Temperature As Variant
Temperature = [{-15.5,-16.4,-17.2,-17.8,-18.5,-
19.1,-20.5,-22.2,-23.7,-25.3,-26.2,-25.1,-22.6,-
20.1,-17.8,-13.8,-13.7,-15.8,-19.3,-21.3,-22.0,-
21.3,-21.8,-21.5}]
Dim Pressure As Variant
Pressure = [{741.44,742.35,743.29,744.20, 744.83,
745.10,745.62,746.20,746.71,747.07,747.32,747.02,746
.54,746.18,745.66,744.97,744.33,743.85,743.24,742.83
,742.13,741.30,740.00,738.92}]
Dim Humidity As Variant
Humidity =
[{94.6,94.8,94.8,94.8,93.9,93.5,93.7,94.8,96.0,
96.5,97.7,95.8,92.7,89.7,80.7,65.4,63.1,70.6,
85.5,91.6,93.1,94.6,94.8,95.0}]
Dim DateX(24) As Date
Dim sd As String
For i = 1 To 24
 sd = "14.01.2014 " & i - 1 & ":00:00"
 DateX(i) = CDate(sd)
Next i

pathx = ActiveWorkbook.path + "\"
Set MyWs = DBEngine.Workspaces(0)
If Dir(pathx + "DBproba1.mdb") <> "" Then Kill pathx
                                    + "DBproba1.mdb"

Set MyDb = MyWs.CreateDatabase(pathx +
      "DBproba1.mdb", dbLangCyrillic, dbVersion40)

strSQL = "Create table weather ([key_weather]
       COUNTER CONSTRAINT key_weather PRIMARY KEY,
       [data_weather] DATETIME,[temperature_weather]
       DOUBLE, [humidity_weather] DOUBLE,
      [pressure_weather] DOUBLE)"
MyDb.Execute strSQL

For i = 1 To 24
  strSQL = "Insert into weather "(data_weather,
          temperature_weather,humidity_weather,
          pressure_weather) values (" &
          Format(DateX(i),"\#mm\/dd\/yyyy
          hh:nn:ss\#") & "," & Replace(CStr
```

```
            (Temperature(i)), ",", ".") & "," &
            Replace(CStr(Humidity(i)), ",", ".") & ","
          & Replace(CStr(Pressure(i)), ",", ".") & ")"
    MyDb.Execute strSQL
  Next i
  ActiveWorkbook.Worksheets("Лист1").Activate
  strSQL = "Select data_weather, temperature_weather
           from weather"
  Set rst = MyDb.OpenRecordset(strSQL, dbOpenSnapshot)
  i = 1
  Do Until rst.EOF()
    Cells(i, 1) = rst!data_weather
    Cells(i, 2) = rst!temperature_weather
    rst.MoveNext
    i = i + 1
  Loop
  rstj.Close
  MyDb.Close
  MyWs.Close
End Sub
```

12.5. Работа с базами данных при помощи ADO (ActiveX Data Object)

Говоря о доступе к базам данных нельзя не упомянуть об *ODBC (Open Database Connectivity)* источниках соединения с базой. Для работы с *ODBC* необходимо выполнить следующие действия:

1. Если это необходимо, то выполнить установку *ODBC* драйвера для вашей *СУБД*. Возможно для этого понадобится установка клиентского ПО на ваш персональный компьютер. Все работы выполняются с правами администратора.

2. В меню «Пуск/Настройка/Панель управления/Администрирование» открыть иконку Источники данных (ODBC) Открыть закладку «Пользовательский DSN» и нажать кнопку «Добавить». В появившемся окне выбрать драйвер Microsoft Access Driver (в зависимости от версии может быть - *.mdb, может быть *.mdb, *.accdb) В поле имя источника данных набрать “con_weather”. В разделе «База данных» нажать кнопку выбрать и в диалоговом окне выбрать базу, например *«...\DBproba1.mdb*» (может быть расширение accdb). Завершив указанные выше операции, нажать кнопку ОК.

3. Перейти в режим редактирования *VBA*. В меню *Tools\References* выбрать библиотеку *Microsoft ActiveX Data Objects 2.8 Library*.

4. Набрать и выполнить следующую программу

```
Sub db_stud()
Dim MyCon As New Connection
Dim StrSQL As String
Dim rs As Recordset

strSQL = "Select data_weather, temperature_weather
          from weather"
MyCon.Open "con_weather"
Set rs = New Recordset
rs.Open StrSQL, MyCon, adOpenForwardOnly,
        adLockReadOnly, adCmdText

ActiveWorkbook.Worksheets("Лист1").Activate
i = 1
Do Until rs.EOF()
   Cells(i, 1) = rs!data_weather
   Cells(i, 2) = rs!temperature_weather
   rs.MoveNext
   i = i + 1
 Loop
 rs.Close
 MyCon.Close
End Sub
```

12.6. Работа с текстовыми файлами

Теперь несколько слов скажем о работе с текстовым файлом, наиболее универсальным хранилищем информации. Может быть текстовые файлы не всегда удобны и надежны для хранения информации, но, т.к. они менее всего подвержены изменению «формата», то могут быть прочитаны и через десятилетия, независимо от версий программ, которые заносили туда информацию.

Для работы с текстовым файлом требуется знать имя файла, путь к файлу, задать файловую переменную или дескриптор файла, объявленную, как длинное целое число (в некоторых случаях для замена переменной - дескриптора используют конструкцию из решетки и целого числа, например, `#1`). Перед открытием файла дескриптор инициализируют, используя функцию `FreeFile`. Обычно программисты `VBA` работают с текстовыми файлами, используя для чтения строк конструкцию `Line Input`, а для записи - оператор `Print`.

Пример чтения/записи из одного текстового файла в другой текстовый файл

```
Sub File_proba1
  Dim path As String
  Dim fNameIn As String
  Dim fNameOut As String
  Dim hFileOut As Long
  Dim hFileIn As Long
  Dim strBuf As String

fNameIn = "Input.txt"
path = ActiveWorkbook.path & "\"
dim ss as string
ss=""
hFileIn = FreeFile
Open path + fNameIn For Input Access Read As
                                          hFileIn
Do Until EOF(hFileIn)
  Line Input #hFileIn, strBuf
  ss= ss+strBuf + vbLF
Loop
Close hFileIn
End sub
```

Пример записи в файл

```
fNameOut="Output.txt"
hFileOut = FreeFile
Open path + fNameOut For Output Access Write As
                                          hFileOut
Print #hFileOut, Trim(strBuf)
Close hFileOut
```

В предыдущем примере при открытии файла на запись, если файл с данным именем существовал, то он должен был перезаписаться, а запись начаться сначала. Если необходимо дописывать информацию в конец файла, то можно воспользоваться следующей конструкцией

```
Sub Add_str_to_txt()
  Dim iText As String
  Dim iFullName As String
  iFullName = "C:\testfile.txt"
  iText="Текст добавляемой строки"
  Open iFullName For Append As #1
      Print #1,iText
      Close #1
End Sub
```

Ниже приведен пример не совсем стандартной работы с файлом, в котором русские сокращения заменяются английские эквиваленты

```
Sub zamena_simvolov()
  Dim s As String, v As Variant, i As Long
  Dim l As Long
  Open "C:\testfile.txt" For Input As #1:
          s = Input(LOF(1), 1): Close #1
  v = Split(s, vbCrLf)
  Open "C:\testfile.txt" For Output As #1
  For i = 0 To UBound(v) - 1
    s = v(i)
    s = Replace(s, "ВБА", "VBA")
    s = Replace(s, "Эксель", "Excel")
    Print #1, s
 Next
 Close #1
End Sub
```

Одна из задач при работе с фалами заключается в определении количества файлов в заданном каталоге, их имен, даты создания и т.п. Приведенный ниже пример того как можно решить данную задачу с использованием объекта *FileSystemObject*, и цикла *For Each ... Next* используемого при работе с коллекциями

```
Sub proba_file()
Set fso = CreateObject("Scripting.FileSystemObject")
path_x = ActiveWorkbook.Path
Set folder_x = fso.GetFolder(path_x)
Set ff = folder_x.Files
ss = ""
ii = 0
For Each f1 In ff
  ss = ss + f1.Name + vbLf
  ii = ii + 1
Next
MsgBox "Обработано " & ii & " файлов " & vbLf & ss

End Sub
```

13. Объектная модель Excel

Работая с приложениями *MS Office* как простой пользователь, так и практикующий программист фактически сталкиваются с классами и объектами «*на каждом шагу*». Приложения, рабочие книги, рабочие листы, формы, кнопки – все это объекты. Ниже мы попытаемся вкратце рассказать что же это

такое, для того, чтобы зная общие закономерности проще находить правильные решения различных прикладных задач.

Класс – это некоторая совокупность однородных объектов, относящихся к определенной сущности.

Объект – экземпляр класса, конкретная реализация сущности, структура, содержащая данные и методы для работы с этими данными.

Свойства *(Property)* - скалярные величины, определяющие различные параметры объекта

Например, свойство *Visible* для таблицы может иметь следующие значения

```
xlSheetVisible (-1)
xlSheetHidden(0)
xlSheetVeryHidden(2)
```

Методы *(Method)* – Способы выполнения действий, доступных для конкретного объекта – набор инструкций для работы с объектом. Эффективное средство выполнения заранее предопределенных действий.

Коллекции *(Collection)* – набор объектов, представленных как отдельный объект. Название коллекции формируется из названия объекта добавлением окончания (s), обозначающего множественное число.

Объекты в коллекции всегда отличаются индексом, всегда имеют свойства *Count* – количество элементов, начинающийся с 1 и метод *Add*

Перебор объектов в коллекции можно осуществить при помощи цикла *For Each ... next*

```
Sub XX()
 Dim  zSheet as WorkSheet
 For Each zSheet in WorkSeets
   MsgBox zSheet.Name
 Next zSheet
End Sub
```

Для объекта *Workbooks* (который является коллекцией) существует свойство *Count* которое содержит сведения о количестве открытых книг

Иерархия объектов

Application
 Workbook
  Worksheet
   Range

Взаимодействие с листом

a) Задание полной иерархии объектов

```
Workbooks("book1.xls")
       .Worksheets("Sheet1").Range("a1").Value=10
Workbooks("book1.xls")
       .Worksheets("Sheet1").Range("a1:a10").Value=10
```

b) Создание переменной

```
Dim w As Workbook
Dim s As Worksheet
Set w = Workbooks("book1.xls")
Set s = w.Worksheets("sheet1")
MsgBox s.Range("a1").value
```

Вызов объектов в иерархии должен быть выполнен в следующем порядке:

```
Application, Workbook, Worksheet, Range
```

В обратном порядке использовать нельзя

Ошибка:

```
Worksheets("Sheet1").Workbooks("book1.xls").save
```

Однако, можно обратиться к родителям

```
Worksheets("Sheet1").Parent.Save
```

Доступ к ячейке листа может быть осуществлена следующим способом:

```
ActiveWorkbook.Worksheets("sheet1").Activate
Cells(1,3).Value =5        ' запись в ячейку листа
MyVar = Cells(2,5).value   ' чтение из ячейки листа
```

Далее покажем примеры использования методов для разных объектов

13.1. Объект APPLICATION

```
MsgBox Application.ActiveCell.Address
MsgBox Application.ActivePrinter
MsgBox Application.ActiveSheet.Cells(11,11).Select
MsgBox Application.ActiveWindow.Caption
MsgBox Application.ActiveWorkbook.Name
```

`AddIns` – дополнительные модули, загруженные в Excel

```
Sub test()
 dim  MyAddIn As AddIn
```

```
  For Each MyAddIn in AddIns
    MsgBox MyAddIn.FullName
  Next MyAddIn
End Sub
```

```
Application.Columns(3).Select
Application.Rows(2).Select
```

Application.MemoryFree
Application.MemoryTotal
Application.MemoryUsed
Application.OperationSystem
Application.Quit

ResentFiles – последние открытые файлы

```
Sub test2()
Dim myFile As RecentFile
p = MsgBox("Show Files? " & Applica-
tion.RecentFiles.Count, vbYesNo, "APP")
If p = vbYes Then
    For Each myFile In Application.RecentFiles
      MsgBox myFile.Name
    Next
End If
End Sub
```

Selection

Application.Selection.Addres – адрес ячейки
Application.Selection.Worksheet.Name

Sheets

Application.Sheets("Sheet1").Print
Application.Sheets("Sheet1").PrintPreview

ThisWorkbook

Application.Undo
Application.UserName
Application.Version

13.2. Объект WORKBOOK

```
Activate
ActiveSheet.Name
Close
```

HasPassword – возвращает true или false в зависимости от того, есть пароль или нет

PrintOut
PrintPreview
ReadOnly
Save
SaveAs
Saved – *сохранены ли изменения*
Sheets
Windows
Worksheets

Приведенный ниже пример демострирует запуск процедуры StartX, находящейся в первом модуле проекта, при загрузке рабочей книги (при открывании файла xls)

```
'MicrosoftExcelObject
'ThisWorkbook

Private Sub Workbook_Open()
  Call Module1.StartX
End Sub
```

13.3. Объект WINDOW

перебор окон: Activate, ActiveteNext, ActivatePrevious

```
Example: Windows(1).Activate,
         ActiveWindow.ActivateNext
```

ActiveCell.Address
ActivePane – свойство окна
Windows(1).ActivePane.VisibleRange.Address – видимая область экрана, адреса ячеек
ActiveSheet.Name
ActiveWindow.Caption
ActiveWindow.Close

Ниже перечислены свойства окна, устанавливаемые при помощи булевых переменных (логического типа данных)

DisplayFormulas
DisplayGridlines
DisplayHeadings
DisplayHorisontalScrollBar
DisplayWorkbookTabs
DisplayZeros

Пример того, как можно спрятать окно с закладками внизу открытого окна

```
ActiveWindow.DisplayWorkbookTabs = False
```

FreezePanes

Закрепление областей в текущей позиции курсора

```
ActiveWindow.FreezePanes = True
```

RangeSelection

```
ActiveWindow.RangeSelection.Address
```

SelectedSheets

```
ForEach MySheet In ActiveWindow.SelectedSheets
 MsgBox MySheet.Name
Next
```

Split – разбить (разделить окно)

```
With ActiveWindow
    .SplitColumn = 2
    .SplitRow = 2
End With
```

Отменить разделение окна

```
ActiveWindow.Split = False
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.Split = False          'method one
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.SplitColumn = 0        'method two
ActiveWindow.SplitRow = 0
```

WindowState

xlMaximazed
xlMinimized
xlNormal
Zoom=80 ' масштаб – 80%

13.4. Объект WORKSHEET

Worksheets("sheet1"). Calculate

CheckSpelling
Comments.Count
Delete
PrintOut
PrintPreview

Protect
Protect("orange")
Range
SaveAs("") ' хотя принадлежит Worksheet, но сохраняет книгу целиком
Select ' выделить отдельный лист

```
SetBackGroundPicture("C:\MyPicture.bmp")
                        'любой рисунок в качестве фона
Unprotect("orange")
Visible = False         ' делает лист невидимым
```

Приведенный ниже пример показывает снятие защиты листа, изменение содержимого ячейки A16 и установка защиты листа.

```
sub pwd_x()
  WorkBookX="MyFile.xls"
  NastrSheet="MyWorkSheet"
PWD="patato"
Workbooks(WorkBookX)
            .Worksheets(NastrSheet).UnprotectPWD)
Workbooks(WorkBookX)
            .Worksheets(NastrSheet).Cells(1, 16) = 0
Workbooks(WorkBookX)
            .Worksheets(NastrSheet).Protect(PWD)
End sub
```

13.5. Объект RANGE

Диапазон ячеек. Можно менять одновременно несколько ячеек на листе

Activate

AddComment 'Добавляет комментарий в ячейку только для одной ячейки, если комментарий уже есть, то будет ошибка

```
Worksheets("Sheet1")
              .Range("A1").AddComment("MyCommtnt")
```

Address

BorderAround(n) 'n – тип линии, рамка вокруг

Calculate 'пересчет значений в ячейках

Cells.Count ' подсчет количества ячеек в диапазоне

CheckSpelling

Clear 'Очищает все: значения, комментарии и форматы в диапазоне ячеек

ClearComments

ClearContents 'удаляет только данные

```
Worksheets("Sheet1").Range("A1").ClearContents
```

ClearFormats

Column, Row ' Номера соответственно первой строки и первого столбца диапазона

Columns, Rows ' Коллекции можно использовать для прохождения по диапазону

Rows.Count ' подсчет рядов

ColumnWidth = m

ColumnHeight= n

Copy PaseteSpecial ' Копирование и специальная вставка

```
Worksheets("Sheet1")
     .Range("A1").PasteSpecial Type:=xlPasteValues
```

PrintPreview

PrintOut 'Для вывода на принтер диапазона

Replace 'Заменяет определенный символ, найденный в диапазоне на другой

```
Worksheets("Sheet1")
                  .Range("A1:B10").Replace "a","b"
```

Select ' программно выделяет как мышью диапазон

Text ' возвращает текст, размещенный в ячейке. Можно использовать только для

'чтения

Value ' можно читать и писать

WrapText=True ' Определяет будет ли текст в ячейке переноситься

14. Работа с пользовательскими формами

Программная автоматизация позволяет решать многие задачи без участия пользователя. Существуют полностью автоматические системы сбора и обработки данных. Однако, большинство информационно-аналитических систем разрабатываются как автоматизированные, т.е. требуют для работы участие человека. Для обеспечения полноценного и удобного взаимодействия пользователя с такой системой используют различные интерфейсы. Наиболее удобным и привычным интерфейсом является интерфейс пользовательских форм.

Обычно под формой понимают окно «контейнер» для размещения элементов управления. Как и любое программное окно, объект формы имеет свои свойства и методы. Ниже перечислены некоторые самые важные свойства форм (кроме ShowModal, все они применимы и для других элементов управления):

- *Name* — определяет имя формы. После создания формы ее имя, предлагаемое по умолчанию (UserForm) рекомендуется заменить на имя, соответствующее выполняемой функции (это относится ко всем элементам управления);
- *Caption* — определяет заголовок формы (по умолчанию совпадает с именем формы);
- *Enabled* — если установлено в False, пользователь работать с формой не сможет. Используется для временного отключения формы;
- *ShowModal* — если установлено в True (по умолчанию), пользователь не может перейти к другим формам или вернуться в документ, пока не закроет эту форму. В версиях до VBA6 поддерживались только модальные формы.

Большая часть основных свойств относится к внешнему виду, размерам и местонахождению окон.

14.1. Самые важные методы форм

В процессе редактирования формы (из окна редактора *Visual Basic*) форму можно запускать по нажатию клавиши <F5>. После того, как форма будет готова, вы должны обеспечить запуск этой формы в документе. Для запуска формы нужно воспользоваться методом *Show()*:

```
UserForm1.Show
```

Если форма уже была загружена в память, она просто станет видимой, если еще нет — то будет автоматически загружена (произойдет событие Load).

Саму эту команду, можно вызвать, например:

- из обычного макроса, привязанного к кнопке или клавиатурной комбинации;
- из автозапускаемого макроса (макроса с названием *AutoExec для Word*);
- из кода для элемента управления, расположенного в самом документе (например, *CommandButton*) или на другой форме — для перехода между формами;
- поместить ее в обработчик события Open для документа Word или книги Excel, чтобы форма открывалась автоматически при открытии документа.

После того, как пользователь введет/выберет нужные данные на форме и нажмет на требуемую кнопку, форму необходимо убрать. Можно спрятать форму (использовать метод Hide()), например:

```
UserForm1.Hide
```

При этом форма будет убрана с экрана, но останется в памяти. Потом при помощи метода Show() можно будет опять ее вызвать в том же состоянии, в каком она была на момент «прятанья», а, можно, например, пока она спрятана, программно изменять ее и расположенные на ней элементы управления. Окончательно форма удалится из памяти при закрытии документа. Если форма больше точно не потребуется, можно ее удалить из памяти при помощи команды Unload:

```
Unload UserForm1
```

Остальные методы, связанные с формой, относятся либо к обмену данными через буфер обмена *(Copy(), Cut(), Paste())*, либо к служебным возможностям формы (*PrintForm(), Repaint(), Scroll()).*

Важнейшая концепция *VBA* — события. Событие (*event*) — 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.

VBA построен таким образом, чтобы создавать на нем программы, управляемые событиями (*event-driven*). Такие программы противопоставляются устаревшему процедурному программированию.

14.2. Самые важные события форм

• *Initialize* — происходит при подготовке формы к открытию (появлению перед пользователем). Обычно в обработчик для этого события помещается код, связанный с открытием соединений базы данных, настройкой элементов управления на форме, присвоение им значений по умолчанию и т.п.

• *Click* (это событие выбирается по умолчанию) и *DblClick* — реакция на одиночный и двойной щелчок мыши соответственно. Для формы это событие используется достаточно редко. Обычно обработчик щелчков используется для кнопок (элементов управления *CommandButton*).

• *Error* — это событие используется при возникновении 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 (например, по команде Unload). Обычно

используется для разрыва открытых соединений с базой данных, освобождения ресурсов, протоколирования и т.п. Если работа формы завершается аварийно (например, запустившее форму приложение выдало команду *End*), то это событие не возникает.

- остальные события связаны либо с изменением размера окон, либо с нажатиями клавиш, либо с активизацией (получением фокуса)/деактивизацией (потерей фокуса).

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.

14.3. Элемент управления CommandButton (кнопка)

Элемент управления *CommandButton* (кнопка) — самый распространенный элемент управления в формах. Главное событие для кнопки, к которому привязывается программный код – это *Click*.

Самые важные свойства кнопки:

- *Cancel* — если для него установить значение True, то это значит, что кнопка будет нажиматься при нажатии на клавишу <Esc>. Как правило, на такие кнопки помещаются надписи типа «Отмена», «Выход», «Вернуться в окно приложения». Однако кроме назначения клавише <Esc>, ничего больше этой кнопке такое свойство не дает. Необходимо будет еще добавить код в обработчик события *Click*.

- *Caption* — надпись, которая будет на кнопке;

- *Default* — такая кнопка будет считаться нажатой, если пользователь нажал на клавишу <Enter>, а фокус находился в другом месте формы (но не на другой кнопке). Обычно такие кнопки являются главными, по которым выполняется действие, ради которого создавалась форма (печать отчета, занесение информации в базу данных, отправка почты и т.п.);

- *Picture* — назначает кнопке вместо надписи определенный рисунок;

- *TakeFocusOnClick* — будет ли передаваться управление этой кнопке при нажатии на нее. По умолчанию *True*.

Ниже рассмотрим несколько примеров обрабатывающих события инициализации и завершения работы с формой

```
Private Sub UserForm_Initialize()
   MsgBox "Hello"
End Sub
Private Sub UserForm_Terminate()
```

```
   MsgBox "Bye"
End Sub
```

Примеры работы с кнопкой и заголовком формы

```
Dim flag_x as boolean
Private Sub CommandButton1_Click()
   if flag_x then
      CommandButton1.Caption  = "Hello!"
      UserForm1.Caption = "My Name is Vasya"
      flag_x= false
   else
     CommandButton1.Caption  = "Bye!"
     UserForm1.Caption = "What is you NAME?"
      flag_x= true
  end if
End Sub

Private Sub CommandButton2_Click()
   Unload UserForm1
End Sub

Private Sub UserForm_Initialize()
MsgBox "Hello Загрузка"
End Sub

Private Sub UserForm_Terminate()
 MsgBox "Bye Завершение"
End Sub
```

14.4. Элемент управления TextBox (Текстовое поле)

Элемент управления *TextBox* используется в следующих случаях

- для приема каких-либо текстовых данных, вводимых пользователем (например, для отправки по почте, для занесения в базу данных и т.п.);
- для вывода пользователю текстовых данных с возможностью их редактирования (из базы данных, листа *Excel* и т.п.);
- для вывода пользователю текстовых данных с возможностью копирования и печати, но без возможности изменения (классический пример — текст соглашения).

Некоторые важные свойства этого элемента управления:

- *Value* (или *Text*, эти два свойства для текстового поля идентичны) — то текстовое значение, которое содержится в этом поле. Ис-

пользуется 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. Использовать не рекомендуется, так как может нарушиться весь дизайн вашей формы.

- *ControlSource* — ссылка на источник текстовых данных для поля. Может ссылаться, например, на ячейку в *Excel*, на поле в *Recordset* и т.п. При изменении пользователем данных в текстовом поле автоматически изменится значение на источнике, определенном в ControlSource.

- *ControlTipText* — текст всплывающей подсказки, которая появляется, когда пользователь наводит указатель мыши на элемент управления. Рекомендуется к заполнению для всех элементов управления (для самой формы не предусмотрена).

- *Enabled* — если переставить в *False*, то текст в поле станет серым и с содержимым поля ничего сделать будет нельзя (ни ввести текст, ни выделить, ни удалить). Обычно это свойство используется (для всех элементов управления), чтобы показать пользователю, что этот элемент управления отключен до выполнения каких-либо условий.

- *Locked* — поле будет выглядеть как обычно, пользователь сможет выделять и копировать данные из него, но не изменять их.

- *MaxLength* — максимальная длина значения, которое можно ввести в поле. Иногда можно использовать свойство *AutoTab* — при достижении определенного количества символов управление автоматически передается другому элементу управления.

- *MultiLine* — можно ли использовать в текстовом поле несколько строк или необходимо обойтись одной. Если вам нужно текстовое поле для приема одного короткого значения, подумайте, нельзя ли вместо него обойтись функцией InputBox.

- *PasswordChar* — указать, за каким символом будут «скрываться» вводимые пользователем значения. Используется, при вводе пароля.

- *ScrollBars* — будут ли показаны горизонтальная и вертикальная полосы прокрутки (в любом сочетании). Если текст может быть большим, без них не обойтись.

- *WordWrap* — рекомендуется включать в тех ситуациях, когда значение MultiLine стоит в True. В этом случае будет производиться автоматический переход на новую строку при достижении границы текстового поля.

Остальные свойства по большей части относятся к оформлению текстового поля и его содержания, а также настройкам редактирования.

Главное событие для текстового поля — это событие Change (то есть изменение содержания поля). Обычно на это событие привязывается проверка вводимых пользователем значений или синхронизация введенного значения с другими элементами управления (например, сделать доступной кнопку, изменить текст надписи и т.п.)

Ниже приведен пример использования текстового поля

```
Private Sub UserForm_Initialize()
UserForm1.Label1.Caption = "Введите Имя"
End Sub

Private Sub CommandButton1_Click()
  strName = UserForm1.TextBox1.Text
  If strName = "" Then
    Label1 = "А где ИМЯ?"
  Else
    TextBox1.Text = "Привет, " & strName
    Label1 = "ОК"
  End If
End Sub

Private Sub CommandButton2_Click()
    TextBox1.Text = ""
End Sub
```

14.5. Элемент управления ComboBox (комбинированный список)

Данный элемент управления используется в тех случаях, когда список позиций для выбора необходимо формировать динамически на основании данных из источника (базы данных и т.п.). Также комбинированный список можно использовать, если пользователю необходимо выбрать одно или несколько значений из списка размером от 4-х до нескольких десятков позиций. Если позиций меньше, то проще использовать переключатели. Если позиций больше четырех, то ориентироваться в списке становится неудобно и необходимо использовать специальные приемы, когда пользователь вводит первые буквы нужного слова и в списке остаются только значения, которые начинаются на эти буквы;

Заполнение списка производится путем использования метода `AddItem()`. Обычно он помещается в обработчик события Initialize для формы. Применение его может выглядеть так:

```
Private Sub UserForm_Initialize()
   ComboBox1.AddItem "Санкт-Петербург"
```

```
    ComboBox1.AddItem "Ленинградская область"
    ComboBox1.AddItem "Москва"
    ComboBox1.AddItem "Московская область"
End Sub
```

Самые важные свойства комбинированного списка:

• *ColumnCount, ColumnWidth, BoundColumn, ColumnHeads, RowSource* — свойства, которые применяются при работе со списками из нескольких столбцов. Обычно эти свойства пользователями не используются (гораздо проще сделать несколько комбинированных списков);

• *MatchEntry* — показывает будут ли при вводе пользователем первых символов значения выбраться подходящие позиции из списка. Возможность очень удобная, рекомендуется сохранить значение по умолчанию.

• *MatchRequired* — разрешается ли пользователю вводить то значение, которого нет в списке. По умолчанию False, то есть разрешено.

• *Value (или Text)* — позволяет программным способом установить выбранное значение в списке или вернуть выбранное/введенное пользователем значение.

Остальные свойства *(AutoSize, Enabled, Locked, ControlText, ControlTipText, MaxLength)* — применяются точно так же, как и для *TextBox*.

Главное событие для комбинированного списка — *Change*, то же, что и для *TextBox*. Обычно в обработчике этого события проверяются введенные пользователем значения, эти значения переносятся в текстовое поле или *ListBox* (если нужно дать пользователю возможность выбрать несколько значений, поскольку свойства *MultiSelect* у *ComboBox* нет) и т.п.

14.6. Элемент управления ListBox (простой список)

Этот элемент управления функционально похож на комбинированный список, однако отличается тем, что все введенные значения сразу видны в поле, аналогичном текстовому, поэтому большое количество позиций в нем уместить трудно. Еще одно отличие заключается в том, что пользователь не может вводить свои значения, ему предоставляется выбор только из готового списка. Однако у этого элемента управления есть и преимущества: в нем пользователь может выбирать не одно значение, как в *ComboBox*, а несколько.

Обычно *ListBox* используется как промежуточный элемент для отображения выбранных пользователем через *ComboBox* значений, либо для вывода небольших списков данных.

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.

14.7. Элементы управления CheckBox (флажок), ToggleButton (кнопка с фиксацией)

Флажки и кнопки с фиксацией используются для выбора вариантов, которые не взаимоисключают друг друга.

Обычно у *CheckBox* используют три свойства:

• *Caption* — надпись справа от флажка, которая показывает, что выбирается этим флажком;

• *TripleState* — если в False (по умолчанию), то флажок может принимать только два состояния: установлен и нет. Если для *TripleState* установить значение *True*, то появляется третье значение: *Null*, изображает установленный серый недоступный флажок.

• *Value* — состояние флажка. Может принимать значения True (флажок установлен), *False* (снят) и *Null* — «серый флажок» (когда свойство *TripleState* установлено в *True*).

ToggleButton выглядит как кнопка, которая при нажатии становится «нажатой», а при повторном нажатии отключается..

14.8. Элементы управления OptionButton (радио-кнопки) и Frame (рамка)

Если *CheckBox* используют, когда нужен выбор из взаимно не исключающих вариантов, то OptionButton нужен для выбора из вариантов взаимно исключающих.

Для данных элементов основными свойствами являются свойства: *Caption* (надпись) и *Value (True* или *False)* – установлен флажок или нет. Главное событие – *Change*.

Выбор должен предоставляться из нескольких переключателей, и при выборе одного из них другой автоматически очищается (переходит в состояние *FALSE*). Если необходимо выбрать из нескольких наборов вариантов, то в этом случае переключатели группируют.

Самый простой вариант группировки — просто использовать форму/вкладку на форме. Если переключатели находятся на одной вкладке, то они автоматически считаются взаимоисключающими. Если же нужно осуществить разбиение на более мелкие группы, то используют элемент управления *Frame*. *Frame* — это просто рамка, которая выделяет прямоугольную область на форме. Помещенные внутрь рамки переключатели считаются взаимоисклю-

чающими. При желании рамку можно сделать невидимой, установив для свойства BorderStyle значение 1 и убрав значение свойства *Caption*.

14.9. Элементы управления ScrollBar (полоса прокрутки) и SpinButton (счетчик)

Полосы прокрутки (*ScrollBars*) чаще всего встречаются в текстовых полях, когда введенный текст полностью не умещается на экране. ScrollBars также могут использоваться в качестве отдельного элемента управления (его еще называют «ползунок»), для осуществления плавного выбора какого-то значения из заданного диапазона (например регулирования уровня громкости или яркости). SpinButton – это «полоса прокрутки» без «полосы». Регулировка производится только нажатием на крайние элементы. *Change* – главное событие для этого элемента управления.

• *Max и Min* — максимальное и минимальные значения, которые можно задать при помощи этого элемента управления. Возможный диапазон — от *-32 767 до +32 767*. При этом максимальное значение вполне может быть меньше минимального — просто ползунок придется тянуть в обратную сторону.

• *LargeChange и SmallChange* — определяет какими шагами будет двигаться ползунок при перемещении (путем щелчка на полосе ниже ползунка или при нажатии на кнопку направления соответственно).

• *Orientation* — определяет расположение ползунка (вертикальное или горизонтальное). По умолчанию для этого свойства установлено значение 1. Это означает, что ориентация определяется автоматически, в зависимости от конфигурации пространства на форме, отведенного элементу управления (что больше — длина или высота). При помощи этого свойства можно также и явно указать вертикальное или горизонтальное расположение ползунка.

• *ProportionalThumb* — определяет размер ползунка: будет ли он пропорционален размеру полосы прокрутки (по умолчанию) или фиксированного размера.

• *Value* — главное свойство этого элемента управления. Определяет положение ползунка и то значение, которое будет возвращать этот элемент управления программе.

Как правило, ползунок применяется вместе с отображением выбранной при помощи него информации. Самый простой вариант применения ползунка показан в примере:

```
Private Sub ScrollBar1_Change()
Label1.Caption = ScrollBar1.Value
End Sub
```

14.10.Элементы управления TabStrip и MultiPage

Оба этих элемента управления применяются в том случае, если элементов управления очень много для того, чтобы уместить их на одной странице формы. При помощи TabStrip и *MultiPage* можно создавать на форме несколько вкладок (страниц), между которыми сможет перемещаться пользователь. Основное отличие между этими элементами управления заключается в том, что на вкладках *MultiPage располагают разные элементы (при переключении закладок вид формы меняется), а на TabStrip* – всегда одинаковые элементы управления (при переключении между вкладками вид формы не меняется, меняется только номер вкладки) Последний вариант может понадобиться, например, для реализации шаблона заполнения разных таблиц, похожими по смыслу полями.

Свойства:

• *MultiRow* — можно ли будет использовать несколько рядов вкладок.

• *TabOrientation* — где будут расположены вкладки (по умолчанию — сверху).

• *Value* — номер вкладки, которая открыта в настоящий момент (нумерация начинается с 0).

Основное событие – *Change* (то есть переход между вкладками). К нему можно привязать. например, проверку уже введенных пользователем значений или выдачу предупреждений.

14.11.Элемент управления Image

Позволяет отобразить на форме рисунок, который будет реагировать на щелчок мышью. В качестве альтернативы можно использовать свойство *Picture* для формы (если нужен фоновый рисунок для всей формы). Также свойства Picture можно использовать для элементов управления *Label* или *CommandButton.* При использовании элемента управления *Image* изображение копируется внутрь документа и внешний его файл становится не нужен. Основное событие *Click.*

Свойства:

• *Picture* — выбрает изображение для формы;

• *PictureAlignment* — выбирает расположение изображения на форме. По умолчанию — по центру;

• *PictureSizeMode* — определяет режим растяжения/уменьшения элемента в случае, если он не соответствует размеру области;

• *PictureTiling* — размножает маленький рисунок, чтобы он покрыл все отведенную ему область («черепица»).

14.12.Дополнительные элементы управления VBA

Выше были рассмотрены стандартные элементы управления, которые изначально помещены в *ToolBox* и доступны для размещения в формах. Очевидно, что возможности форм этим не ограничиваются. В принципе можно использовать тысячи элементов управления, встроенных в Windows, или поставляемых отдельно. Для того, чтобы разместить дополнительные элементы в *ToolBox,* надо щелкнуть правой кнопкой мыши по пустому пространству формы ToolBox и выбрать пункт *Additional Controls* (дополнительные элементы управления), после этого можно выбрать нужный элемент. При этом необходимо помнить, применение нестандартных элементов управления требует наличие на компьютере, на котором будет использоваться желаемая форма, всех необходимых библиотек.

В качестве достаточно полезных «нестандартных» элементов управления можно упомянуть следующие элементы:

Microsoft Web Browser (Internet Explorer). Если его поместить на форму или на рабочий лист Excel, а в обработку события нажатия на CommandButton поместить следующий текст: *(WebBrowser1.Navigate "http:\\www.google.com.ru")* то в окне браузера появится изображение искомого Интернет ресурса. В этот же элемент управления можно выводить информацию и из локальной *Intranet* сети с адреса *LocalHost* (с текущего компьютера). Такая возможность может существенно расширить функциональность пользовательской формы.

Если надо подключить к компьютеру какие-то внешние устройства, например, термометр, систему управления гирляндами, систему управления вентилятором и т.п., то можно воспользоваться передачей данных по последовательному интерфейсу, который может быть реализован при помощи элемента MsComm (последовательный коммуникационный порт)

Выбор даты удобнее осуществлять при помощи элемента *Calendar*. Главное его свойство – это *Value*

Элемент *RefEdit* похож на текстовое поле с кнопкой в правой части. При нажатии на эту кнопку форма, на которой размещен этот элемент управления, «спрячется», а пользователю будет предоставлена возможность выбрать одну ячейку *Excel* или диапазон ячеек. После того, как пользователь завершит выбор, он опять вернется в окно формы, а в *RefEdit* будет помещена информация об адресе выбранного диапазона. Такой же адрес, конечно, можно вводить и вручную. Главное свойство этого элемента управления — *Value*.

15. Отладка программ и обработка ошибок

При работе с программой достаточно часто возникают ошибки, причем ошибки разного рода и уровня сложности. При возникновении ошибки появля-

ется окно сообщений, предлагающее прервать или продолжить выполнение программы. При прерывании программы у программиста появляется возможность перейти в то место программы где возникла ошибка и посмотреть значения переменных алгоритм и т.п. При необходимости исследовать работу программы до места возникновения ошибки, можно поставить точку останова либо щелкнув курсором мыши по левому бордюру окна редактирования программы напротив строки в которой требуется прервать выполнение программ, либо это можно сделать из окна отладки. После остановки работы программы можно запустить ее пошаговое выполнение. Основные команды пошагового выполнения указаны в окне отладчика *Debug (Setp Into, Step Over, Step Out, Run to Cursor)*

Для обеспечения отладочного вывода информации необходимо открыть отладочное окно View\Immediate Vindow (Ctrl-G) и воспользоваться конструкцией *Debug.Print*

```
Debug.Print("Hello")
```

Также можно вывести необходимую информации о значениях переменных или элементов массивов, использую опцию отладчика Debug\Add Watch

В некоторых случаях, если есть желание не «травмировать психику» пользователя, программист старается избежать возникновения стандартного окна сообщения об ошибке, да еще и на английском языке. Для решения этой задачи можно написать свой обработчик ошибок, используя команду *On Error Go To metka*. При этом надо перед меткой обязательно поставить команду *Exit Sub* или *Exit Function*, чтобы правильно работающая программа или функция никогда не перешла к обработке несуществующей ошибки.

При возникновении ошибки интерпретатор просто перейдет на метку за которой и будет написана программа, обрабатывающая все ошибки. Под обработкой ошибки можно понимать какое-то действие программиста, выполняемое при возникновении той или иной ошибки. Это может быть простое сообщение от том, что «все нормально» и программа решила завершить работу на сегодня. Может быть просто интерпретация английского описания ошибки на русском языке.

При обработке ошибок часто используют либо свойство *Number* объекта *Err*, либо свойство Description того же объекта

```
Sub proba_err()
  On Error GoTo metka:
  .....
  a= 10/0  'ошибочны оператор
  'следующий оператор
  ......
  Exit Sub
  metka:
```

```
  MsgBoks "Произошла ошибка: " & Err.Number & VbLf & "
  означающая " & Err.Description
End Sub
```

Можно сделать так, чтобы любая ошибка игнорировалась: `On Error Go To 0`

Оператор Resume передает управление в ту строку, где была совершена ошибка. Можно его использовать только в том случае, если ошибка устранена

Можно сделать так, чтобы после вашего сообщения об ошибки выполнение программы было продолжена со следующего за «ошибочным» оператора (со следующей строки) `On Error Resume Next`

```
sub err_example1()
 On Error GoTo metka:
 .....
 ошибочны оператор
 следующий оператор
 ......
 Exit Sub
 metka:
 MsgBoks "Ошибка"
 Resume Next
End Sub
```

Если оператор `On Error Resume Next` поставить в начале программы, то просто ошибочные операторы на будут выполняться, а управление будет передаваться следующему за ним оператору

Чтобы после обработки ошибки перейти к определенной точке в программе (`metka1`), можно использовать конструкцию `Resume metka1`

`Err.Clear` – очищает объект `Err` от предыдущей ошибки

`Err.Raise num` – позволяет имитировать (вызывать) ошибку с номером num

Ниже приведен пример имитации и обработки ошибки переполнения

```
Sub ErrExample()
  Dim Msg
  On Error Resume Next
  Err.Clear
  Err.Raise 6
  If Err.Number <> 0 Then
     Msg = "Ошибка # " & Str(Err.Number) & " была
  сгенерирована " _
              & Err.Source & Chr(13) & Err.Description
```

```
        MsgBox Msg, , "Сообщение об ошибке",
    Err.HelpFile, Err.HelpContext
    End If
End Sub
```

16. Использование диаграмм для построения графиков

Чтобы построить график функции, визуально отобразить другие зависимости в Excel пользуются диаграммами *(меню Вставка | Диаграмма)*. Диаграмма по-английски называется графиком *(Chart)* и ему соответствует объект *Chart*. В объектной модели *Excel* предусмотрен также и объект Diagram, который является схемой отношений и который мы рассматривать не будем.

Для того, чтобы в Excel с использованием VBA построить график (диаграмму) сначала объявляем объект типа *Chart*:

```
Dim dChart As Chart
```

Создание диаграммы производится путем вызова метода *Add()* коллекции `Charts`:

```
Set dChart = ActiveWorkbook.Charts.Add(, ActiveSheet)
```

Чтобы она обрела содержание, необходимо выполнить еще несколько действий

Первое (и единственное обязательное действие) — определить источник данных для диаграммы, для чего предназначен метод `SetSourceData()`. В качестве источника может выступать только объект Range (он передается в качестве первого и единственного обязательного параметра этого метода). Второй параметр (необязательный) определяет, в каком порядке считывать данные — сначала по столбцам, потом по строкам или наоборот. Например, в нашем случае это может выглядеть так:

```
dChart.SetSourceData Sheets("Лист1").Range("A1:A10")
```

Чтобы диаграмма обрела содержание, необходимо определить ее тип (по умолчанию она будет выглядеть как «обычная гистограмма», т. е. ряд из столбцов разной длины). Для определения типа используется свойство `ChartType`, для которого разработчиками предусмотрено 73 значения. Например, чтобы преобразовать диаграмму в обычный график, можно использовать код вида:

```
dChart.ChartType = xlLineMarkers
```

Еще одно действие, которое чаще всего необходимо совершить – это добавить дополнительные ряды на диаграмму. Для этой цели необходимо создать и получить ссылку на объект `Series` — ряд, а потом для ряда определить свойство `Values` (ему передается в качестве значения объект Range):

```
Dim dSeries As Series
Set dSeries = dChart.SeriesCollection.NewSeries
```

```
dSeries.Values = Worksheets("Лист1").Range("B1:B10")
```

По умолчанию диаграмма создается в оперативной памяти и помещается на отдельный лист. Если нам необходимо поместить ее на уже существующий лист, то в этом случае ее надо создать вначале на отдельном листе, а затем переместить при помощи метода *Location*. Отдельный лист, созданный для диаграммы, при этом автоматически исчезнет:

```
dChart.Location xlLocationAsObject, "Лист1"
```

Необходимо обратить внимание, что метод Location принимает в качестве первого параметра одну из констант *(xlLocationAsNewSheet* — переместить на специально создаваемый новый лист, *xlLocationAsObject* — переместить на объект, т. е. на лист), а в качестве второго — не объект листа, а его имя. Для избежания ошибок предпочтительнее получать имя листа программным образом. Проблема с методом *Location,* заключается в том, что после перемещения диаграммы внутрь листа объектная ссылка на эту диаграмму теряется, и надо находить объект этой диаграммы заново. При попытке повторного обращения к объекту *Chart* выдается сообщение «*Automation Error*». Поэтому лучше всего вызов метода Location помещать в самый конец кода, посвященного диаграмме. В противном случае придется разыскивать созданную диаграмму и заново получать на нее объектную ссылку, например, так:

```
Dim dSeries As Series
Set dSeries = _
Worksheets(1).ChartObjects(1)
              .Chart.SeriesCollection.NewSeries
oSeries.Values = Worksheets(1).Range("B1:B10")
```

Теперь рассмотрим пример построения диаграмм вцелом.

```
Sub diagr()
  Dim dChart As Chart
  Set dChart = ActiveWorkbook
               .Charts.Add(, ActiveSheet)
  dChart.SetSourceData
                Sheets("Лист1").Range("A1:A10")
  dChart.ChartType = xlLineMarkers
  Dim dSeries As Series
  Set dSeries = dChart.SeriesCollection.NewSeries
  dSeries.Values = Worksheets("Лист1")
                .Range("B1:B10")
   dChart.Location xlLocationAsObject, "Лист1"
End Sub
```

Остальные многочисленные параметры диаграммы настраиваются при помощи свойств и методов объектов *Chart.*

ChartArea — возвращает объект *ChartArea*, который представляет собой область, занимаемую диаграммой, и используется для настройки внешнего вида диаграммы (свойства *Font, Interior* и т. п.). Если настраивается внешний вид только той части диаграммы, которая используется непосредственно для вывода графика, используется свойство PlotArea. По умолчанию диаграмма размещается по центру листа. Если необходимо ее переместить в точно определенное место листа, используются свойства *Top, Height, Left* и *Width* объекта *ChartArea*.

ChartTitle — возвращает одноименный объект, при помощи которого можно настроить заголовок диаграммы (с такими свойствами, как *Text, Font, Border* и т. п.).

ChartType — важнейшее свойство, которое определяет тип диаграммы.

HasDataTable — если это свойство имеет значение True, то в нижней части диаграммы (по умолчанию) появится таблица с числами, на основе которых была создана диаграмма. Одновременно будет создан программный объект *DataTable*, при помощи которого можно будет настроить представление этой таблицы. Схожим образом работают свойства H*asLegend*, *HasPivotFields и HasTitle*.

Name — это свойство позволяет настроить имя диаграммы (как название вкладки в Excel). По умолчанию диаграммы называются последовательно «*Диаграмма1*», «*Диаграмма2*» и т. п.

SizeWithWindow — если поставить значение этого свойства в *True*, то размер диаграммы будет подогнан таким образом, чтобы точно соответствовать размеру листа. По умолчанию установлено в *False.*

Tab — свойство которое позволяет настроить при помощи одноименного объекта внешний вид вкладки в книге Excel для диаграммы (или просто листа). Например, чтобы пометить вкладку зеленым цветом, можно воспользоваться кодом:

```
oChart.Tab.Color = RGB(0, 255, 0)
```

Visible — позволяет спрятать диаграмму без ее удаления.

Далее представлены наиболее важные методы объекта *Chart.*

Activate() — используется очень часто. Он позволяет сделать диаграмму активной (т. е. просто перейти на нее).

ApplyCustomType() — позволяет создать диаграмму своего собственного пользовательского типа (для этого необходимо вначале создать шаблон для этого типа и поместить его в галерею).

ApplyDataLabels() — позволяет поместить на диаграмму метки для размещенных на ней данных. Этот метод принимает множество параметров, которые позволяют настроить отображение меток (показывать или не показывать значения и т. п.).

Axes() — возвращает объект, представляющий оси диаграммы. Затем этот объект можно использовать для настройки данных осей.

Copy() — позволяет скопировать диаграмму в другое место книги (например, для создания новой диаграммы на основе существующей). Для переноса существующей диаграммы в другое место можно воспользоваться методами *Location() или Move()*.

CopyPicture() — метод, который позволяет поместить диаграмму в буфер обмена как изображение. Затем это изображение можно вставить в документ Word или в любое другое место. Еще один вариант — воспользоваться методом Export(), который позволяет создать рисунок, представляющий диаграмму, в виде файла на диске.

Delete() — удаляет диаграмму.

Evaluate() — как обычно, этот метод позволяет найти нужную диаграмму в книге по ее имени.

PrintOut() — отправляет диаграмму на печать. Этот метод принимает множество параметров, которые позволяют настроить такой вывод.

Refresh() — позволяет обновить диаграмму, если изменились данные, на основе которых она строилась.

Select() — выделяет диаграмму (равносильно щелчку по ней левой кнопкой мыши). Обратный метод *Deselect()* снимет выделение (равносильно нажатию *<Esc>)*.

SetBackgroundPicture() — позволяет «подложить» под диаграмму фоновый рисунок. Конечно, он должен быть не очень ярким.

SetSourceData() — важнейший метод, который позволяет определить данные, на основе которых строится диаграмма. Про него мы уже говорили.

Выше мы показали, как построить так называемую диаграмму категорий, в которой ось X – это множество натуральных чисел, начиная с *1*. При построении графика функций очень часто требуется отдельным множеством задавать значения области определения *(оси X)* и отдельным множеством область значений *(ось Y)*. Пример такого построения показан ниже. Обратите внимание, что ось X задается *параметром XValues* объекта *SeriesCollection*.

```
Sub xxy()
Dim dChart As Chart
```

```
Dim dSeries As Series

Set dChart = ActiveWorkbook.Charts.Add(, ActiveSheet)
dChart.ChartType = xlXYScatterLines
Set dSeries = dChart.SeriesCollection.NewSeries
dSeries.XValues = Worksheets("Лист1").Range("A1:A10")
dSeries.Values = Worksheets("Лист1").Range("B1:B10")
dChart.Location xlLocationAsObject, "Лист1"

End Sub
```

17. Использование функций Windows API

Иногда при программировании на языке *VBA* возникает необходимость использовать функции *Windows API* (*Application Programming Interface*). Эти функции предоставляют огромное количество возможностей от различных вариантов отображения окон и кнопок до организации сетевого взаимодействия. Всего насчитывается порядка тысячи таких функций.

Для использования API функций их надо сначала объявить с использованием инструкции

Declare [Public | Private] Declare Function Имя Lib "Библиотека" [Alias "Псевдоним"] [([аргументы])] [as имя типа]

Эта инструкция размещается в блоке объявлений модуля. Если ключевое слово «*Alias*» не используется, то «*Имя*» в объявлении функции должно полностью совпадать с именем API функции. «*Библиотека*»– имя библиотеки, в которой находится используемая *API* функция. Если используется ключевое слово «*Alias*», то «Псевдоним» должен полностью совпадать с *именем API* функции, а имя объявляемой функции может быть любым. Аргументы и имя типа объявляются точно так же, как и для обыкновенной функции *VBA*.

Ниже приведен пример объявления и использования *API* функции *Sleep*. 'Для использования *API функции*, как было сказано выше, в начале модуля до функций и процедур объявляют библиотечную функцию:

```
Private Declare Sub Sleep Lib "kernel32" (ByVal
dwMilliseconds As Long)
```

Затем в любой процедуре вызываем объявленную функцию Sleep

```
Sleep 1000
```

18. Организация задержки и работа с таймером

В предыдущем примере мы показали как организовать задержку, используя *API интерфейс*. Ниже вы сможете увидеть, каким образом аналогичная задача решается с использованием «*внутренних*» ресурсов *VBA*.

Один из способов – использование метода *Wait* уровня *Application*

```
Application.Wait (Now + TimeValue("0:00:01"))
' задержка 1 секунда
```

Оба предыдущих варианта организации задержки (*Sleep, Wait*) имеют существенный недостаток – они не дают пользователю работать во время ожидания. Использование таймера, по большому счету, нельзя назвать задержкой, однако во многих случаях программисту нужна не задержка между событиями, а организация вызова последовательности событий через определенные интервалы, что по большому счету – одно и то же. С последней задачей, особенно если требуется организовать не единичные, а циклически повторяющиеся временные интервалы между событиями, замечательно справляется таймер.

Ниже приведем простой пример работы двух таймеров. В данном примере интерфейсной частью программы является пользовательская форма UserForm3, запускающаяся процедурой ppx в немодальном режиме и имеющая пять кнопок («*Start Clock1», «Stop Clock1», «Start Clock2*», «*Stop Clock2», «Exit*»). По нажатию кнопки «*Start Clock1*» запускается процедура *tick*, которая записывает текущее время в ячейку *A1* первого листа рабочей книги и запускает саму себя в момент времени *nextTime* (через одну секунду) при помощи функции *Application.OnTime*. Такие перезапуски продолжаются до тех пор, пока не нажата кнопка «*Stop Clock1*», которая запускает процедуру *End_Tick*, отменяющую установленное расписание для процедуры OnTime. При нажатии «*Start Clogk2*» запускается процедура *Tick1* с временем перезапуска 3 скунды, выводящая текущее время в заголовок формы.

Перед запуском примера рекомендуется установить формат ячейки *A1* рабочего *листа1* в следующем виде: *ДД.ММ.ГГГГ ч:мм:сс*

```
Sub ppx()
 UserForm3.Show (0)
End Sub


Sub tick()
 ThisWorkbook.Worksheets(1).Cells(1, 1) = Now
 nextTime = Now + #12:00:01 AM# '
 Application.OnTime nextTime, "tick"
End Sub
```

```
Public Sub tick1()
 UserForm3.Caption = Now
 nextTime1 = Now + #12:00:03 AM# '
 Application.OnTime nextTime1, "tick1"
End Sub

Sub tick1_end()
 On Error Resume Next
 Application.OnTime nextTime1, "tick1", , False
End Sub

Sub tick_end()
 On Error Resume Next
 Application.OnTime nextTime, "tick", , False
End Sub
```

Обработчики нажатий на кнопки пользовательской формы должны иметь следующий вид

```
Private Sub CommandButton1_Click()
  Call Module1.tick
End Sub

Private Sub CommandButton2_Click()
   Call Module1.tick_end
End Sub

Private Sub CommandButton3_Click()
   Call Module1.tick1
End Sub

Private Sub CommandButton4_Click()
   Call Module1.tick1_end
End Sub

Private Sub CommandButton5_Click()
   Unload UserForm3
End Sub
```

19. Полезные функции

В заключение хотелось бы рассказать еще о нескольких полезных функциях

DoEvents() – эта функция позволяет передать управление операционной системе на время выполнения какой-то длительной операции, чтобы обработать накопившиеся в операционной системе события. После этого продолжение операции VBA продолжается.

Environ() — возвращает абсолютный путь для переменных окружения (в командной строке эта же задача реализуется выполнением команды *SET)*. Если нужно, например, получить путь ко временному каталогу, то можно выполнить следующую команду:

```
MsgBox Environ("TEMP")
```

GetAllSettings() — позволяет получить (в виде двумерного массива) из реестра (ветвь HKEY_CURRENT_USERS) все параметры, которые относятся к указанному приложению. Функция *SaveSetting()* позволяет записать информацию в реестр, а *DeleteSetting()* — удалить. *GetSetting()* позволяет получить информацию об определенном параметре.

```
Dim MySettings As Variant, intSettings As Integer
```

' Помещает значения некоторых параметров в реестр

```
SaveSetting appname:="MyApp", section:="Startup", _
Key:="Top", setting:=75
SaveSetting "MyApp", "Startup", "Left", 50
```

' Выборка параметров

```
Debug.Print GetSetting(appname := "MyApp", section :=
"Startup", key := "Left", default := "25")
MySettings = GetAllSettings(appname:="MyApp",
                        section:="Startup")
For intSettings = LBound(MySettings, 1) To
                        UBound(MySettings, 1)
   Debug.Print MySettings(intSettings, 0),
                        MySettings(intSettings, 1)
    Next intSettings
 DeleteSetting "MyApp", "Startup"
End Sub
```

Shell() — позволяет запускать из *VBA* внешнюю программу. Эту функцию можно использовать для запуска любых внешних программ из вашего приложения. Однако, рекомендуется применение специальных объектов *WshShell* и *WshExec* из библиотеки *Windows Script Host Object*

Model. С их помощью можно передавать в окно клавиатурные комбинации, принимать и передавать значения через командную строку и т. п.

TypeName() — функция возвращает имя типа данных для переданной переменной. Обычно используют для определения типа данных для значений, получаемых из базы данных или путем вызова метода какого-то объекта.

20. Приложение 1. Примеры решения типовых задач

Написать функцию *Func_Z*, которая принимает три целых числа в качестве параметров и возвращает целое число равное удвоенной сумме переданных параметров. Запустить функцию *Func_Z* из процедуры *Proba_2*. Слагаемые в процедуре должны вводится при помощи окна InputBox. Результат должен быть выведен при помощи окна *MsgBox*

```
'*** K1 ***
Function Func_Z(a As Integer, b As Integer, c As In-
teger) As Integer
  Func_Z = 2 * (a + b + c)
End Function
Sub Proba_2()
  Dim X As Integer, Y As Integer, Z As Integer
  X = InputBox("Введите первое слагаемое")
  Y = InputBox("Введите второе слагаемое")
  Z = InputBox("Введите третье слагаемое")
  MsgBox "Сумма равна: " & Func_Z(X, Y, Z)
End Sub
```

Написать функцию Func_X3, которая принимает в качестве параметров одну булеву переменную и три целых числа и возвращает целое число равное удвоенной сумме переданных целочисленных параметров, если первый параметр равен TRUE. Во всех других случаях функция должна возвращать -1. Функцию Func_X3 необходимо вызвать из процедуры Proba_2. Значения параметров должны вводиться в процедуре при помощи окна InputBox. Значение логической переменной должно быть введено в текстовом режиме словами «ИСТИНА», «ЛОЖЬ». При вводе должна осуществляться проверка адекватности введенных данных требуемым условиям. Результат должен быть выведен при помощи окна MsgBox

'---

```
'*** K2 ***
Function Func_X3(a As Boolean, c As Integer, d As In-
teger, e As Integer)
   Func_X3 = -1
   If a Then Func_X3 = 2 * (c + d + e)
End Function
Sub Proba_21()
```

```
Dim xa As Boolean, xb As Integer, xc As Integer, xd
As Integer, ss As String
ss = InputBox("Введите логическую переменную", "Окно
ввода", "Истинна")
If ss = "Истинна" Then
  xa = True
Else
  If ss = "Ложь" Then
    xa = False
  Else
    MsgBox "Введены не очень хорошие данные" & vbLf &
"До свидания!"
    Exit Sub
  End If
End If
xb = InputBox("Введите целое число для первого пара-
метра")
xc = InputBox("Введите целое число для второго пара-
метра")
xd = InputBox("Введите целое число для третьего пара-
метра")
MsgBox "Результат равен:" & Func_X3(xa, xb, xc, xd)

End Sub
```

Написать функцию вычисляющую утроенную сумму двух целых чисел, которые передаются в функцию качестве параметров. Используя стандартный вызов функции с рабочего листа рабочей книги MS Excel, обеспечить вычисление утроенных сумм чисел, помещенных в ячейки «A13» и «B17» рабочего листа «Лист1». Результат вычислений должен помещаться в ячейку «C22»

```
'*** К3 ****
Function Fun_z2(a As Integer, b As Integer)
                                        As Integer
  Fun_z2 = 3 * (a + b)
End Function
```

Вручную создать текстовый файл input.txt, в первую строку через пробел записать два числа. На языке VBA написать программу, считывающую первое число в переменную X, второе число – в переменную Y. Вычислить Z=X2+Y-14. Результат записать в ячейку B12 листа Result того же файла, в котором написана программа

'

```
' *** K4 ***
Sub proba3()
 Dim InFileName As String
 Dim hFile As Long
 Dim X As Integer, Y As Integer, Z As Integer
 Dim pathx As String, StrBuf As String
 Dim VarBuf() As String

 pathx = ThisWorkbook.Path + "\"
 InFileName = "input.txt"
 If Dir(pathx + InFileName) = "" Then
   MsgBox "Не могу найти файл: " & pathx + InFileName
   Exit Sub
 End If
 hFile = FreeFile
 Open pathx + InFileName For Input Access Read As
#hFile
 Line Input #hFile, StrBuf
 VarBuf = Split(StrBuf, " ")
  X = VarBuf(0)
   Y = VarBuf(1)
  Z = X * 2 + Y - 14
 ThisWorkbook.Worksheets("Result").Range("B12") = Z
End Sub
```

В текстовый файл MMZ1.txt вручную через пробел занести четыре двузначных числа. Написать процедуру Z2X(), которая открывает, текстовый файл с именем, выбираемым из элемента управления ComboBox (в список имя файла MMZ1.txt должно быть занесено при инициализации формы) После открытия из файла должны быть считаны искомые четыре двузначных числа, вычислена их утроенная сумма, а результат выведен в выходной файл с именем RESULT1.txt и в окно сообщений MsgBox

'

'******** К5 ********

' Программа в Форме (Рис.20.1)

'**********

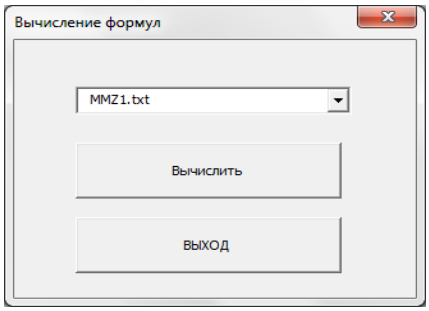

Рис.20.1

```
Private Sub CommandButton1_Click()
 Call Z2X
End Sub
Private Sub CommandButton2_Click()
   Unload UserForm1
End Sub
Private Sub UserForm_Initialize()
 ComboBox1.AddItem "MMZ1.txt"
 ComboBox1.AddItem "MMZ2.txt"
 ComboBox1.AddItem "MMZ3.txt"
 ComboBox1.AddItem "MMZ4.txt"
 ComboBox1.ListIndex = 0
End Sub
Sub Z2X()
Dim InFileName As String
Dim OutFileName As String
Dim mas(4) As Integer
Dim ss As Integer
Dim Z As Integer
Dim hFile As Long
Dim hFileOut As Long
Dim pathx As String, StrBuf As String
Dim VarBuf() As String
 pathx = ThisWorkbook.Path + "\"
 InFileName = ComboBox1.Value
 OutFileName = "RESULT1.txt"
 If Dir(pathx + InFileName) = "" Then
   MsgBox "Не могу найти файл: " & pathx + InFileName
   Exit Sub
 End If
```

```
 hFile = FreeFile
 Open pathx + InFileName For Input Access Read As
#hFile
 Line Input #hFile, StrBuf
 Close #hFile
 VarBuf = Split(StrBuf, " ")
 ss = 0
 For i = 0 To 3
   ss = ss + CInt(VarBuf(i))
 Next i
 Z = ss * 3
  hFile = FreeFile
  Open pathx + OutFileName For Output Access Write As
#hFile
  Print #hFile, CStr(Z)
  Close #hFile
 MsgBox "Результат равен: " & Z
End Sub
```

На заданном рабочем листе заданной рабочей книги создать таблицу заказов (Рис.20.2), содержащую информацию о пяти товарах: Название товара, количество, цена, единицы измерения, сумма. Написать макрос, который выдает отчет о сделанных покупках с использованием окна MsgBox Отчет должен формироваться на основании информации, содержащейся в таблице заказов.

'

'*** К6 ***

	A	B	C	D	E	F
1	Нум п\п	Название	Количество	Цена	Единицы	Сумма
2	1	Молоко	17	47,00р.	л.	799,00р.
3	2	Сметана	10	200,00р.	л.	2 000,00р.
4	3	Творог	3	150,00р.	кг.	450,00р.
5	4	Хлеб	10	25,00р.	булка	250,00р.
6	5	Сыр	5	350,00р.	кг.	1 750,00р.

Рис.20.2

```
Sub proba_K6()
 Dim t_name As String, t_kol As Double
 Dim t_price As Double, t_ed As String
 Dim t_sum As Double
```

```
 Dim str_buf As String
 ThisWorkbook.Worksheets("Лист3").Activate
 For i = 0 To 4
   t_name = Cells(2 + i, 2).Value
   t_kol = Cells(2 + i, 3).Value
   t_price = Cells(2 + i, 4).Value
   t_ed = Cells(2 + i, 5).Value
   t_sum = t_kol * t_price
   Cells(2 + i, 6).Value = t_sum
   str_buf = "Мы купили " & t_kol & " " & t_ed & " "
& t_name & " по " & t_price & " руб. на сумму " &
t_sum & " руб."
   MsgBox str_buf
 Next i
End Sub
```

С использованием MS Excel создать таблицу логарифмов для чисел от одного до 48 в файле рабочей книги. Таблица должна состоять из шести столбцов (аргумент, значение, аргумент, значение, аргумент, значение), строки заголовка и 16 строк данных. Написать макрос автоматически заполняющий данную таблицу. Основание логарифма и лист, куда будет выведен результат необходимо получить из элементов управления ComboBox1 и ComboBox2 пользовательской формы UserForm1. Начальные значения в эти элементы управления необходимо занести при инициализации пользовательской формы

'

'*** К7 ***

'*** часть программы в форме (Рис.20.3) ***

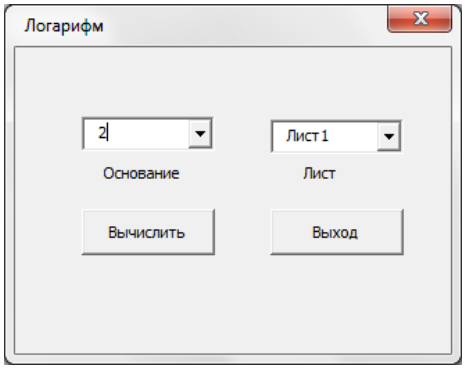

Рис.20.3

```
Public okx As Boolean
Private Sub CommandButton1_Click()
 okx = True
 UserForm4.Hide
```

```
End Sub

Private Sub CommandButton2_Click()
 okx = False
 UserForm4.Hide
End Sub

Private Sub UserForm_Initialize()
  ComboBox1.Clear
  ComboBox2.Clear
  okx = False
  For i = 2 To 7
    ComboBox1.AddItem CStr(i)
  Next i
  ComboBox1.ListIndex = 0
  For i = 1 To 3
    ComboBox2.AddItem "Лист" & CStr(i)
  Next i
  ComboBox2.ListIndex = 0
End Sub
Sub proba_K7()
Dim LBase As Integer
Dim SheetX As String
UserForm4.Show
If UserForm4.okx Then
   LBase = UserForm4.ComboBox1.Value
   SheetX = UserForm4.ComboBox2.Value
   ThisWorkbook.Worksheets(SheetX).Activate
   Range("A1:F50").ClearContents
   MsgBox "Начинаем заполение таблицы"

    For i = 1 To 48
      Select Case i
        Case 1 To 16: j = 1 '0
        Case 17 To 32: j = 3 '16
        Case 33 To 48: j = 5 '32
      End Select
     Cells(i - 8 * (j - 1), j).Value = i
     Cells(i - 8 * (j - 1), j + 1).Value = Log(i) /
Log(LBase)
    Next i
End If
 MsgBox "До свидания!"
End Sub
```

Создать процедуру Proba_X1, которая запускается при открытии рабочей книги и в окне InputBox запрашивает имя студента. Если имена: Василий, Елена, Михаил или Семен, то в окне сообщения должно появиться поздравление с началом зимней сессии. Если имена: Мария, Марина, Маргарита и Самуил, то в окне сообщения должно появиться поздравление с окончанием зимней сессии. В остальных случаях должно быть сообщение: «Добро пожаловать».

```
'
'*** K8 ***
Sub Proba_X1()

 Dim StudName As String
 Dim StrStr As String
 StudName = InputBox("Введите имя студента!")
 Select Case StudName
  Case "Василий", "Елена", "Михаил", "Семен"
    StrStr = "Поздравляем с началом зимней сессии"
  Case "Мария", "Марина", "Маргарита", "Самуил"
    StrStr = "Поздравляем с окончанием зимней сессии"
  Case Else
   StrStr = "Добро пожаловать!"
 End Select
   MsgBox StrStr

End Sub
```

21. Приложение 2. Условия типовых задач

K-1

Тип: ВЫЧИСЛЕНИЕ ФОРМУЛЫ (элементарное). ВХОДНЫЕ данные InputBox РЕЗУЛЬТАТ в MsgBox

(Темы: объявление переменных; арифметические операции; InputBox; MsgBox

---

K-11

Написать функцию Func_Z, которая принимает три целых числа в качестве параметров и возвращает целое число равное удвоенной сумме переданных параметров. Запустить функцию Func_Z из процедуры Proba_2. Слагаемые в процедуре должны вводится при помощи окна InputBox. Результат должен быть выведен при помощи окна MsgBox

K-12

Написать функцию ZZZ, которая принимает четыре целых числа в качестве параметров и возвращает целое число равное удвоенной сумме первых двух чисел за вычетом утроенной суммы вторых двух чисел, которые были переданы в качестве параметров. Слагаемые должны вводится при помощи окна InputBox. Результат должен быть выведен при помощи окна MsgBox

K-13

Написать функцию Func_33, которая принимает два числа с плавающей точкой в качестве параметров и возвращает число с плавающей точкой, равное удвоенной сумме переданных параметров, деленной на 33. Слагаемые должны вводится при помощи окна InputBox. Результат должен быть выведен при помощи окна MsgBox

K-14

Написать функцию Func_Z1, которая принимает два целых числа и одно дробное в качестве параметров и возвращает число равное удвоенной сумме переданных параметров. Запустить функцию Func_Z1 из процедуры Proba_24. Слагаемые в процедуре должны вводится при помощи окна InputBox. Результат должен быть выведен при помощи окна MsgBox

---

K-21

Тип: ВЫЧИСЛЕНИЕ ФОРМУЛЫ (усложненное). ВХОДНЫЕ данные InputBox РЕЗУЛЬТАТ в MsgBox

(Темы: объявление переменных; арифметические операции; InputBox; MsgBox, потоки управления программой If … Then, сравнение строк)

K-22

Написать функцию Func_X3, которая принимает в качестве параметров одну булеву переменную и три целых числа и возвращает целое число равное удвоенной сумме переданных целочисленных параметров, если первый параметр равен TRUE. Во всех других случаях функция должна возвращать -1. Функцию Func_X3 необходимо вызвать из процедуры Proba_2. Значения параметров должны вводиться в процедуре при помощи окна InputBox. Значение логической переменной должно быть введено в текстовом режиме словами «ИСТИНА», «ЛОЖЬ». При вводе должна осуществляться проверка адекватности введенных данных требуемым условиям. Результат должен быть выведен при помощи окна MsgBox

K-23

Написать функцию F_X1, которая принимает в качестве параметров две дробные переменные и одну булеву переменную и возвращает целое число равное удвоенной сумме, переданных дробных параметров поделенной на три, если логический параметр равен FALSE. Во всех других случаях функция должна возвращать -1111. Запустить функцию F_X1 из процедуры Proba_33. Значения параметров должны вводится в процедуре при помощи окна InputBox. Значение логической переменной должно вводится в текстовом режиме словами «ИСТИНА», «ЛОЖЬ». При вводе должна осуществляться проверка адекватности введенных данных. Результат должен быть выведен при помощи окна MsgBox

---

К-3

Тип: ВЫЧИСЛЕНИЕ ФОРМУЛЫ (рабочий лист). ВХОДНЫЕ данные ячейки рабочего листа РЕЗУЛЬТАТ в ячейки рабочего листа

(Темы: объявление переменных; функция; арифметические операции; формула на рабочем листе)

К-31

Написать функцию вычисляющую утроенную сумму двух целых чисел, которые передаются в функцию качестве параметров. Используя стандартный вызов функции с рабочего листа рабочей книги MS Excel, обеспечить вычисление утроенных сумм чисел, помещенных в ячейки «A13» и «B17» рабочего листа «Лист1». Результат вычислений должен помещаться в ячейку «C22»

К-32

Написать функцию вычисляющую утроенную сумму, деленную на два трех дробных чисел, которые передаются в функцию качестве параметров. Используя стандартный вызов функции с рабочего листа рабочей книги MS Excel, обеспечить вычисление искомых сумм чисел, помещенных в ячейки «A13» , «B17» и C19 рабочего листа «Лист2». Результат вычислений должен помещаться в ячейку «C17» рабочего листа «Лист1»

К-4

Тип: ВЫЧИСЛЕНИЕ ФОРМУЛЫ (файл) ВХОДНЫЕ данные в файле txt (xls) РЕЗУЛЬТАТ в файл txt (xls)

(Темы: объявление переменных; работа с текстовым файлом [определение пути к файлу; проверка наличия файла; открывание файла; чтение из файла]; арифметические операции; доступ к ячейкам листа; работа со строками [разбиение строки на подстроку – функция split]; одномерные массивы;)

К-41

Вручную создать текстовый файл input.txt, в первую строку через пробел записать два числа. На языке VBA написать программу, считывающую первое число в переменную X, второе число – в переменную Y. Вычислить $Z=X^2+Y-14$. Результат записать в ячейку B12 листа Result того же файла, в котором написана программа

K-42

Вручную создать текстовый файл input1.txt, в первую строку через пробел записать три числа. На языке VBA написать программу, считывающую первое число в переменную X1, второе число – в переменную X2, третье число – в переменную X3 . Вычислить $Z=X1^2+X2-14*X3$. Результат записать в ячейку B1 листа Res того же файла, в котором написана программа

K-43

Вручную создать текстовый файл input.txt, в первую строку через пробел записать три числа, во вторую строку – через два пробела еще три числа. На языке VBA написать программу, считывающую числа из первой строки в массив X(3), числа из второй строки – в массив Y(3). Вычислить $Z=Y(2)*X(0)^2+Y(0)-14*X(2)$. Результат записать в файл, имя которого записано на в ячейке B12 листа Nastr того же файла, в котором написана программа

K-44

Вручную создать текстовый файл inp.txt, в первую строку через три пробела записать четыре числа, во вторую строку – через два пробела еще четыре числа. На языке VBA написать программу, считывающую числа из первой строки в массив X(4), числа из второй строки – в массив Y(4). Вычислить Z=Y(2)*X(0)2+Y(0)-14*X(2)+2*X(3)+Y(1). Результат записать в файл, имя которого записано на в ячейке A2 листа Лист1 того же файла, в котором написана программа

K-45

На языке VBA написать программу, считывающую число из ячейки A1 листа Лист2 в переменную X, число из ячейки B3 того же листа – в переменную Y. Вычислить Z=X2+Y*43+15. Результат записать в файл output.txt

K-46

На языке VBA написать программу, считывающую число из ячейки B3 листа Sheet2 в переменную X, число из ячейки B3 того же листа – в переменную Y. Вычислить Z=X2+Y*3+15. Результат записать в файл output.txt

K-47

В текстовый файл MMZ.txt вручную через пробел занести три двузначных числа. Написать процедуру ZX(), которая открывает, текстовый файл с именем, получаемым из стандартного диалогового окна открытия файла (MMZ.txt), считывает искомые три числа, вычисляет их утроенную сумму и выводит результат в выходной файл с именем RESULT.txt и в окно сообщений MsgBox

K-48

В текстовый файл ZZ.txt вручную через пробел занести три двузначных числа. Написать процедуру ZR(), которая открывает, текстовый файл с именем, получаемым из стандартного диалогового окна открытия файла (ZZ.txt), считывает искомые три числа, вычисляет их утроенную сумму, деленную на два и выводит результат в выходной файл с именем res.txt и в окно сообщений MsgBox

---

K-5

Тип: ВЫЧИСЛЕНИЕ ФОРМУЛЫ (файл, форма) . ВХОДНЫЕ данные в файле txt, в списке ComboBox РЕЗУЛЬТАТ в файл txt (xls) и MsgBox

(Темы: объявление переменных; работа с текстовым файлом [получение пути к файлу; проверка наличия файла; открывание файла; чтение из файла; запись в файл]; работа с элементом управления ComboBox и пользовательской формой; работа со строками [разбиение строки на подстроку – функция split]; арифметические операции; MsgBox)

K-51

В текстовый файл MMZ1.txt вручную через пробел занести четыре двузначных числа. Написать процедуру Z2X(), которая открывает, текстовый файл с именем, выбираемым из элемента управления ComboBox (в список имя файла MMZ1.txt должно быть занесено при инициализации формы). После открытия из файла должны быть считаны искомые четыре двузначных числа, вычислена их утроенная сумма, а результат выведен в выходной файл с именем RESULT1.txt и в окно сообщений MsgBox

K-52

В текстовый файл M.txt вручную через пробел занести три двузначных числа. Написать процедуру ZX(), которая открывает, текстовый файл с именем, выбираемым из элемента управления ComboBox (в список имя файла M.txt должно быть занесено при инициализации формы). После открытия из файла должны быть считаны искомые три двузначных числа, вычислена разность первых двух чисел, умноженная на третье число, а результат выведен в выходной файл с именем RESULT13.txt и в окно сообщений MsgBox

K-53

Открыть текстовый файл, в котором записана одна строка, содержащая два числа и еще одна строка, содержащая одно число. Считать эти три числа, вычислить их утроенную сумму, вывести результат в выходной файл и в окно сообщений MsgBox. Имя выходного файла должно быть задано с использованием процедуры InputBox

K-54

Написать функцию Func_Z, которая принимает три числа с плавающей точкой в качестве параметров и возвращает целое число равное целой части удвоенной суммы переданных параметров. Запустить функцию Func_Z из процедуры Proba_2. Слагаемые должны вводится при помощи окна InputBox. Результат должен быть выведен при помощи окна MsgBox и записан в ячейку A17 рабочего листа. Имя листа необходимо запросить с использованием элемента управления TextBox1 формы UF1

K-55

Написать функцию Function_Z1, которая принимает два числа с плавающей точкой в качестве параметров и возвращает целое число равное целой части утроенной суммы переданных параметров. Запустить функцию Function_Z1 из процедуры Proba_1. Слагаемые должны вводится при помощи окна InputBox. Результат должен быть выведен при помощи окна MsgBox и записан в ячейку B21 рабочего листа. Имя листа необходимо запросить с использованием элемента управления TextBox1 формы UserF1

K-6

Тип: ТАБЛИЦА ЗАКАЗОВ. ВХОДНЫЕ данные ячейки рабочего листа РЕЗУЛЬТАТ в MsgBox

(Темы: объявление переменных; цикл For; доступ к ячейкам листа; конкатенация строк)

K-61

На заданном рабочем листе заданной рабочей книги создать таблицу заказов, содержащую информацию о пяти товарах: Название товара, количество, цена, единицы измерения, сумма. Написать макрос, который выдает отчет о сделанных покупках с использованием окна MsgBox Отчет должен формироваться на основании информации, содержащейся в таблице заказов.

K-62

На заданном рабочем листе заданной рабочей книги создать таблицу заказов, содержащую информацию о пяти товарах написать макрос, который в виде периодически появляющихся сообщений выдает отчет о сделанных покупках. Отчет должен формироваться на основании информации, содержащейся в таблице. При формировании отчета необходимо использовать цикл FOR и окно стандартного вывода MsgBox

K-63

На заданном рабочем листе заданной рабочей книги создать таблицу заказов, содержащую информацию о пяти товарах: Название товара, количество, цена, единицы измерения, сумма. Написать макрос, который выдает отчет о сделанных покупках с использованием элемента TextBox пользовательской формы. После вывода последней строки отчета форма должна закрыться. Отчет должен формироваться на основании информации, содержащейся в таблице заказов.

K-64

На заданном рабочем листе заданной рабочей книги создать таблицу заказов, содержащую информацию о пяти товарах: Название товара, количество, цена, единицы измерения, сумма. Написать макрос, который выдает отчет о сделанных покупках с использованием элемента ListBox1 пользовательской формы. Название листа, на который помещена форма необходимо получить из элемента ComboBox1 той же формы. После вывода последней строки отчета форма должна закрыться. Отчет должен формироваться на основании информации, содержащейся в таблице заказов. Содержимое в ComboBox должно быть внесена при инициализации формы.

K-7

Тип: ТАБЛИЦА ЛОГАРИФМОВ. ВХОДНЫЕ данные формула РЕЗУЛЬТАТ в лист MS Excel

(Темы: объявление переменных; цикл For; доступ к ячейкам листа; арифметические операции; управление потоком выполнения программы IF ... Then)

K-71

С использованием электронных таблиц MS Excel создать таблицу логарифмов по основанию два для чисел от одного до 64 в файле рабочей книги. Таблица должна состоять из четырех столбцов (аргумент, значение, аргумент, значение), строки заголовка и 32 строк данных. Написать макрос автоматически заполняющий данную таблицу

K-72

С использованием электронных таблиц MS Excel создать таблицу логарифмов по основанию три для чисел от одного до 60 в файле рабочей книги. Таблица должна состоять из шести столбцов (арумент, значение, аргумент, значение, аргумент, значение), строки заголовка и 20 строк данных. Написать макрос автоматически заполняющий данную таблицу

К-73

С использованием электронных таблиц MS Excel создать таблицу логарифмов для чисел от одного до 64 в файле рабочей книги. Таблица должна состоять из четырех столбцов (аргумент, значение, аргумент, значение), строки заголовка и 32 строк данных. Написать макрос, автоматически заполняющий данную таблицу. Основание логарифма и лист, куда будет выведен результат необходимо получить из элементов управления TextBox1 и TextBoxt2 пользовательской формы UF3

К-74

С использованием MS Excel создать таблицу логарифмов для чисел от одного до 48 в файле рабочей книги. Таблица должна состоять из шести столбцов (аргумент, значение, аргумент, значение, аргумент, значение), строки заголовка и 16 строк данных. Написать макрос, автоматически заполняющий данную таблицу. Основание логарифма и лист, куда будет выведен результат необходимо получить из элементов управления ComboBox1 и ComboBox2 пользовательской формы UserForm1. Начальные значения в эти элементы управления необходимо занести при инициализации пользовательской формы

К-8

Тип: ПОЗДРАВЛЕНИЕ ВЫБОРОЧНОЕ. ВХОДНЫЕ данные InputBox РЕЗУЛЬТАТ MsgBox

(Темы: объявление переменных; InputBox; MsgBox; управление потоком выполнения программы If ... Then, Select ... Case)

К-81

Создать процедуру Proba_X1, которая запускается при открытии рабочей книги и в окне InputBox запрашивает имя студента. Если имена: Василий, Елена, Михаил или Семен, то в окне сообщения должно появиться поздравление с началом зимней сессии. Если имена: Мария, Марина, Маргарита и Самуил, то в окне сообщения должно появиться поздравление с окончанием зимней сессии. В остальных случаях должно быть сообщение: «Добро пожаловать».

К-82

Создать процедуру Proba_X1, которая запускается при открытии рабочей книги и в окне InputBox запрашивает имя студента. Затем, если сегодня четное число, то в окне сообщения должно появиться персональное поздравление студента с *окончанием* летней сессии, если число нечетное, то поздравление должно быть с окончанием сессии.

K-83

Создать процедуру Proba_Z3, которая запускается при открытии рабочей книги и в окне InputBox сначала запрашивает имя студента, а затем его фамилию. В итоге программа должна в окне сообщения MsgBox по имени и фамилии поздравить студента с окончанием летней сессии, если его фамилия отличается от Иванов или Иванова. В противном случае необходимо просто поздороваться

K-84

Создать процедуру Proba_X1, которая запускается при открытии рабочей книги, просит ввести имя пользователя, сравнивает его с пятью именами, хранящимися в начальных строках столбца А Листа1. Если введенное имя совпадает, то в окне сообщения выводится текущая дата, время и поздравление с окончанием летней сессии, в противном случае выводится просто поздравление с началом сессии.

K-85

Создать процедуру PP_X2, которая запускается при открытии рабочей книги, просит ввести фамилию пользователя, сравнивает ее с пятью фамилиями, хранящимися в начальных строках столбца С Листа2. Если введенная фамилия совпадает, то в окне сообщения выводится текущая дата, в противном случае выводится просто поздравление с удачным началом дня.

K-9

Тип: ПОЗДРАВЛЕНИЕ (форма). ВХОДНЫЕ ДАННЫЕ элементы управления пользовательской формы ВЫХОДНЫЕ ДАННЫЕ элементы управления пользовательской формы

(Темы: объявление переменных; пользовательская форма; элементы управления; преобразование типов; вычиление временных промежутков между датами)

K-91

Создать пользовательскую форму; Поместить на нее элементы управления Label1, Label2 и CommandButton1. Написать процедуру, обрабатывающую нажатие на кнопку, и выводящую поздравление с началом зимней сессии в элемент управления Label2, а в элемент управления Label1 – количество дней до начала зимней сессии.

K-92

Создать пользовательскую форму; Поместить на нее элементы управления Label1, TextBox1 и CommandButton1; Написать процедуру, обрабатывающую нажатие на кнопку, и выводящую поздравление с окончанием летней сессии в элемент управления Label1, а в элемент управления TextBox1 – количество дней до конца летней сессии. Окно TextBox1 должно быть не доступно для редактирования

K-93

Создать процедуру Proba_new, которая запускается при открытии рабочей книги и через элемент управления TextBox1 просит ввести любое целое положительное число больше нуля (должна быть реализована проверка условий). Процедура должна вывести приветствие с наступлением месяца чей порядковый номер совпадает с остатком от деления введенного числа на 12 в элемент управления TextBox2

K-10

Тип: НАХОЖДЕНИЕ МАКСИМАЛЬНОГО (МИНИМАЛЬНОГО) ЭЛЕМЕНТА МАССИВА. ВХОДНЫЕ ДАННЫЕ в МАССИВЕ результат в MSGBOX

(Темы: объявление переменных; инициализация массива; цикл For ... Next; управление потоком выполнения программы If ... Then; передача параметров по ссылке)

K-101

Написать функцию F_MAX, получающую в качестве параметра по ссылке массив целочисленных элементов и возвращающую значение максимального элемента из данного массива. Написать процедуру, в которой необходимо задать массив из десяти элементов: 3, 7, 21, 11, 4, 4, 67, 8, 91, 1 и вызвать функцию F_MAX для данного массива. Результат вывести в MsgBox

K-102

Написать функцию F_MIN, получающую в качестве параметра по ссылке массив целочисленных элементов и возвращающую значение минимального элемента из данного массива. Написать процедуру, в которой необходимо задать массив из десяти элементов: 33, 17, 121, 131, 4, 44, 67, 18, 91, 11 и вызвать функцию F_MIN для данного массива. Результат вывести в MsgBox

---

К-11

Тип: СОРТИРОВКА ЭЛЕМЕНТОВ МАССИВА. ВХОДНЫЕ ДАННЫЕ в МАССИВЕ результат в MSGBOX

(Темы: объявление переменных; инициализация массива; цикл For ... Next; управление потоком выполнения программы If ... Then; передача параметров по ссылке)

К-111

Написать функцию F_SORT, получающую в качестве параметра по ссылке массив целочисленных элементов и возвращающую значение максимального элемента из данного массива. Результат сортировки остается в том же массиве. Написать процедуру, в которой необходимо задать массив из десяти элементов: 3, 71, 1, 11, 54, 4, 67, 8, 91, 1 и вызвать функцию F_SORT для данного массива. Результат сортировки вывести в MsgBox

К-112

Написать функцию F_SORT1, получающую в качестве параметра по ссылке массив дробных элементов и возвращающую значение минимального элемента из данного массива. Результат сортировки остается в том же массиве. Написать процедуру, в которой необходимо задать массив из десяти элементов: 33.11, 1.134, 1, 0.1, 55, 4.7, 6.7, 8, 9.1, 1 и вызвать функцию F_SORT1 для данного массива. Результат сортировки вывести в MsgBox

---

К-12

Тип: ГЕНЕРАЦИЯ МАССИВА (по начальному элкменту, количесту элементов и шагу) ВХОДНЫЕ ДАННЫЕ в ФОРМЕ результат в MSGBOX

(Темы: объявление переменных; работа с формой и элементами управления; одномерные динамические массивы; циклы; окно сообщений; суммирование элементов массива; арифметические операции)

К-121

Написать процедуру, которая получает в качестве первого параметра значение начального элемента массива MMas, в качестве второго параметра - количество элементов этого массива, с использованием элементов управления TextBox , Label, Command Button, размещенных на форме, запрашивает и вводит шаг изменения значений элементов массива, генерирует массив, вычисляет среднее арифметическое элементов массива и выводит в столбик элементы массива и среднее арифметическое с использованием MsgBox. Примечание: при написании процедуры необходимо сначала объявить массив, заполнить его значениями, затем выполнить необходимые вычисления

К-122

Написать процедуру, которая получает в качестве первого параметра количество элементов массива Mas, в качестве второго параметра – значение начального элемента массива, запрашивает с использованием InputBox шаг изменения значений элементов массива, вычисляет среднее арифметическое элементов массива, выводит в окно сообщения объекта MsgBox значения всех элементов массива, искомое среднее арифметическое и записывает его в ячейку C1 рабочего листа «Лист1»

К-123

Написать процедуру, которая получает в качестве первого параметра количество элементов массива MasMas, в качестве второго параметра – значение начального элемента массива, запрашивает с использованием InputBox шаг изменения значений элементов массива, вычисляет среднее арифметическое элементов массива, выводит в окно сообщения объекта MsgBox значения всех элементов массива, искомое среднее арифметическое и записывает его в ячейку C1 рабочего листа «Лист1»

К-124

Написать процедуру, которая получает в качестве первого параметра количество элементов массива Mas, в качестве второго параметра – значение начального элемента массива, запрашивает с использованием InputBox шаг изменения значений элементов массива, вычисляет среднее арифметическое элементов массива, выводит в окно сообщения объекта MsgBox значения всех элементов массива, искомое среднее арифметическое и записывает его в ячейку C1 рабочего листа, название которого должно быть введено через элемент управления TextBox пользовательской формы UF4

К-13

Тип: ВЫЧИСЛЕНИЕ ХАРАКТЕРИСТИК ДВУМЕРНОГО МАССИВА (файл xls). ВХОДНЫЕ данные в файле xls РЕЗУЛЬТАТ в MsgBox

(Темы: объявление переменных; работа с файлом xls [открывание файла]; двумерные массивы; циклы; доступ к ячейкам листа; окно сообщений; суммирование элементов массива)

К-131

Вручную создать файл zachet.xls; добавить лист *matrix;* в ячейки с B10 по D12 листа *matrix* Записать матрицу 11 2 -4

5 -11 16

21 3 19

Написать на языке VBA процедуру, которая должна находится в файле zachet2.xls Задача процедуры: открыть файл zachet.xls; считать с листа matrix22 матрицу в массив A; вывести в окно сообщений элементы главной диагонали матрицы и сумму этих элементов.

К-132

Вручную: создать файл zet.xls; добавить лист *mix1;* в ячейки с A10 по C12 листа *mix1* записать матрицу 11 42 -4

-25 7 16

11 3 19

Написать на языке VBA процедуру, которая должна находится в файле zet1.xls. Задача процедуры: открыть файл zet.xls; считать с листа mix1 матрицу в массив A; Вывести в окно сообщений элементы побочной диагонали матрицы и сумму этих элементов.

К-14

Тип: ВЫЧИСЛЕНИЕ ХАРАКТЕРИСТИК ДВУМЕРНОГО МАССИВА (файл txt). ВХОДНЫЕ данные в файле txt РЕЗУЛЬТАТ в MsgBox

(Темы: объявление переменных; работа с файлом txt [открывание файла]; работа со строками (функция split); двумерные массивы; циклы; окно сообщений; сравнение элементов массива)

К-141

Вручную: создать текстовый файл matr.txt; Записать матрицу

11 2 -4 7

5 -11 16 21

21 3 19 11

1 31 9 6

Числа должны следовать через два пробела. Написать на языке VBA процедуру, которая должна открыть файл matr.txt; считать из него матрицу в массив А; вывести в окно сообщений строку максимальных элементов, по одному для каждого столбца.

К-142

Вручную: создать текстовый файл matr2.txt; Записать матрицу

11 72 4 22

5 -11 1 8

21 56 9 11

31 31 9 3

Числа должны следовать через два пробела. Написать на языке VBA процедуру, которая должна открыть файл matr2.txt; считать из файла матрицу в массив В; вывести в окно сообщений строку минимальных элементов, по одному для каждого столбца матрицы.

К-15

Тип: РАБОТА С КОЛЛЕКЦИЕЙ РАБОЧИХ КНИГ MS EXCEL. ВХОДНЫЕ ДАННЫЕ коллекция рабочих книг выходные данные MSGBOX

(Темы: объявление переменных; коллекция рабочих книг; цикл for each; окно сообщений; операция по модулю)

К-151

Вручную скопировать в текущий рабочий каталог семь любых файлов созданных в MS EXEL и затем открыть их (Если таких файлов нет, то создать их и «сохранить как» в текущем рабочем каталоге). Затем вручную переименовать один из файлов в «MyZachet.xls». В данном файле написать процедуру, которая закрывает все файлы, кроме от текущего (того, в котором хранится выполняемая процедура). Список имен закрытых файлов вывести в MsgBox в два столбца

К-152

Вручную скопировать в текущий рабочий каталог семь любых файлов созданных в MS EXEL и затем открыть их (Если таких файлов нет, то создать их и «сохранить как» в текущем рабочем каталоге). Затем вручную переименовать один из файлов в «ZZt.xls» В данном файле написать процедуру, которая закрывает все файлы, кроме от текущего (того, в котором хранится выполняемая процедура). Список имен закрытых файлов вывести в MsgBox в три столбца

К-16

Тип: РАБОТА С ФАЙЛОВОЙ СИСТЕМОЙ. ВХОДНЫЕ ДАННЫЕ файлы в заданном каталоге ВЫХОДНЫЕ данные ЛИСТ MS EXEL

(Темы: объявление переменных; объект «файловая система»; коллекция файлов в каталоге; цикл for each; доступ к ячейкам листа; функция split; динамический массив; работа с файлом MS EXCEL [открывание файла]; управление потоком выполнения программы IF ... THEN; сравнение строк)

К-161

В текущем рабочем каталоге должно быть не менее пяти файлов, из них два файла MS EXEL. Необходимо разработать функцию, которая возвращает строку со списком всех файлов, находящихся в текущем каталоге. Имена файлов в возвращаемой строке должны быть разделены пробелами. Используя данную функцию, вывести список файлов, которые находятся в текущем каталоге в первый столбец первого листа (имя каждого файла в отдельную ячейку). Файлы MS EXEL должны быть открыты.

К-162

В текущем рабочем каталоге должно быть не менее семи файлов. Необходимо разработать функцию, которая возвращает строку со списком всех файлов, находящихся в каталоге. Полный путь к данному каталогу должен передаваться в функцию в качестве параметра. Имена файлов в возвращаемой строке должны быть разделены пробелами. Используя данную функцию, вывести список файлов, которые находятся в заданном каталоге во вторую строку первого листа (имя каждого файла в отдельную ячейку). Имена файлов MS EXCEL должны быть выведены в MsgBox, после чегофайлы должны быть открыты

К-17

Тип: КАЛЬКУЛЯТОР (работа с формой). ВХОДНЫЕ ДАННЫЕ элементы управления формы ВЫХОДНЫЕ ДАННЫЕ элементы управления формы

(Темы: объявление переменных; пользовательская форма; элементы управления; арифметические операции; управление потоком выполнения программы IF ... THEN; преобразование типов)

К-171

Создать пользовательскую форму. Поместить на нее три элемента управления TextBox. Поместить на нее два элемента управления CommandButton; Изменить надписи на кнопках на знаки, соответствующие арифметическим операциям сложения и вычитания. Написать программу, обрабатывающую события нажатия на кнопки, выполняющую соответствующие арифметические действия над числами, помещенными в TextBox1 и TextBox2 и выводящую результат операции в TextBox3

К-172

Создать пользовательскую форму. Поместить на нее три элемента управления TextBox поместить на нее два элемента управления CommandButton; Изменить надписи на кнопках на знаки, соответствующие арифметическим операциям сложения и деления. Написать программу, обрабатывающую события нажатия на кнопки, выполняющую соответствующие арифметические действия над числами, помещенными в TextBox1 и TextBox2 и выводящую результат операции в TextBox3. При записи в TextBox1 должны высвечиваться знаки $, а в TextBox2 – знаки ?

К-173

Создать пользовательскую форму. Поместить на нее три элемента управления TextBox. Поместить на нее четыре элемента управления CommandButton. Изменить надписи на кнопках на знаки, соответствующие арифметическим операциям. Написать программу, обрабатывающую события нажатия на кнопки, выполняющую соответствующие арифметические действия над числами, помещенными в TextBox1 и TextBox2 и выводящую результат операции в TextBox3

К-174

Создать пользовательскую форму. Поместить на нее три элемента управления TextBox Поместить на нее два элемента управления CommandButton; Изменить надписи на кнопках на знаки, соответствующие арифметическим операциям сложения и вычитания Написать программу, обрабатывающую события нажатия на кнопки, выполняющую соответствующие арифметические действия над числами, помещенными в TextBox1 и TextBox2 и выводящую результат операции в TextBox3. После нажатия любой кнопки пользовательской формы в заголовке формы должно отображаться текущее время

К-18

Тип: РАСКРАСКА ТЕКСТА. ВХОДНЫЕ ДАННЫЕ рабочий лист MS EXCEL ВЫХОДНЫЕ ДАННЫЕ рабочий лист MS EXCEL

(Темы: объявление переменных; доступ к ячейкам рабочего листа; цикл for ... each; параметры объекта Text; объект Range; управление потоком выполнения программы If ... Then)

К-181

Путем анализа записанного автоматически макроса, протоколирующего ручные действия оператора, получить код программы, изменяющей цвет текста в ячейках рабочего листа MS Excel. Используя эту информацию, написать процедуру, меняющую цвет записанного в ячейках A22:C30 текста, с красного на зеленый.

К-182

Путем анализа записанного автоматически макроса, протоколирующего ручные действия оператора, получить код программы, изменяющей цвет текста в ячейках рабочего листа MS Excel. Используя эту информацию, написать процедуру, меняющую цвет записанного в ячейках A22:C30 текста, с красного на зеленый, в том случае, если там записаны дробные числа.

К-183

Путем анализа записанного автоматически макроса, протоколирующего ручные действия оператора, получить код программы, изменяющей цвет заливки, ячеек. Используя полученную информацию, написать процедуру, меняющую цвет заливки ячеек B22:C34, в которых записаны числа, большие двух и меньшие пяти с прозрачного на красный

К-184

Путем анализа записанного автоматически макроса, протоколирующего ручные действия оператора, получить код программы, изменяющей цвет текста в ячейках рабочего листа MS Excel. Используя эту информацию, написать процедуру, меняющую цвет записанного в ячейках A22:C30 текста, с красного на зеленый, в том случае, если там записаны числа больше пяти.

К-185

На Листе1 рабочей книги вручную создать таблицу, состоящую из четырех столбцов и семи строк. В первый столбец таблицы занести произвольные неповторяющиеся названия школьных предметов (уроков) в количестве семи шт. Ячейки остальных трех столбцов заполнить произвольным образом числами от двух до пяти (оценками). Написать процедуру, которая раскрашивает двойки в красный цвет, тройки – в желтый, четверки – в синий, пятерки оставляет без изменения.

К-186

На Листе2 рабочей книги вручную создать таблицу, состоящую из трех столбцов и пяти строк. Левая верхняя ячейка таблицы: B2. В первый столбец таблицы занести произвольные неповторяющиеся названия школьных предметов (уроков) в количестве семи штук. Ячейки остальных трех столбцов заполнить произвольным образом числами от двух до пяти (оценками). Написать процедуру, которая раскрашивает двойки в красный цвет, тройки – в синий, четверки и пятерки оставляет без изменения.

К-19

Тип: СИСТЕМЫ СЧИСЛЕНИЯ. ВХОДНЫЕ ДАННЫЕ строка из цифр ВЫХОДНЫЕ ДАННЫЕ MSGBOX

(Темы: объявление переменных; цикл For ...Next; управление потоком выполнения программы If ... Then; работа со строками; арифметические операции)

К-191

Написать функцию PREOBR1, принимающую в качестве параметра строковую переменную и возвращающую целое десятичное число. Строка, переданная в качестве параметра состоит из нулей и единиц и представляет собой двоичное число, старшие разряды, которого расположены в начале строки. Необходимо в функции PREOBR1 реализовать преобразование данного двоичного числа в десятичное. Функция должна проверять двоичное или не двоичное число введено.

K-192

Написать функцию PREOBR2, принимающую в качестве параметра строковую переменную и возвращающую целое десятичное число. Строка, переданная в качестве параметра, состоит из нулей и единиц и двоек и представляет собой троичное число, старшие разряды, которого расположены в начале строки. Необходимо в функции PREOBR1 реализовать преобразование данного троичного числа в десятичное. Функция должна проверять правильность введенного числа

K-20

Тип: ЭНТРОПИЯ ПО ШЕННОНУ. ВХОДНЫЕ ДАННЫЕ рабочий лист MS EXCEL. ВЫХОДНЫЕ ДАННЫЕ MSGBOX

(Темы: объявление переменных; цикл For ...Next; доступ к ячейкам листа; арифметические операции)

K-201

В ячейки A1 записано количество символов алфавита обитателей звездной системы MM-DEMC. В последующих ячейках данного столбца находится информация о том, сколько раз соответствующий символ встретился в тексте гимна данной звездной системы. Написать процедуру, которая выводит значение энтропии по Шеннону текста гимна, если в столбец A, начиная с A1, записаны следующие числа: 4, 77, 32, 11, 3. Результат вывести в MsgBox

K-202

В ячейки B3 записано количество символов алфавита жителей острова X-EMC в нижних ячейках данного столбца записано сколько раз соответствующий символ встретился в приветственной речи президента острова. Написать процедуру, которая выводит значение энтропии по Шеннону текста приветствия, если в столбец B, начиная с B3, записаны следующие числа: 6, 7, 32, 11, 39, 345, 99. Результат вывести в MsgBox

---

К-21

Тип: КОЛИЧЕСТВО ИНФОРМАЦИИ. ВХОДНЫЕ ДАННЫЕ Форма. ВЫХОДНЫЕ ДАННЫЕ Форма

(Темы: объявление переменных; формы; элементы управления; арифметические операции)

К-211

Создать пользовательскую форму, содержащую четыре элемента ComboBox, пять элементов Lable и один CommandButton. При инициализации элементы Label1, Label2, Label3, Label4 должны принять значения: «Номер этажа», «Номер зала», «Номер стеллажа», «Номер полки»; в элементы ComboBox с 1 по 4 должны быть добавлены числа: 1-7, 1-3, 1-10, 1-8, и выбраны значения по умолчанию; При нажатии на кнопку CommandButton1 должна быть выполнена процедура, вычисляющая количество информации, содержащееся в сообщении о том, что книга находится на выбранном этаже, в выбранном зале, на выбранном стеллаже, и на выбранной полке. Результат должен быть выведен в Label5

К-212

Создать пользовательскую форму, содержащую три элемента ComboBox, четыре элемента Lable и один CommandButton. При инициализации элементы Label1, Label2, Label3 должны принять значения: «Номер зала», «Номер стеллажа», «Номер полки»; в элементы ComboBox с 1 по 3 должны быть добавлены числа: 1-3, 1-11, 1-9, и выбраны значения по умолчанию; При нажатии на кнопку CommandButton1 должна быть выполнена процедура, вычисляющая количество информации, содержащееся в сообщении о том, что книга находится в выбранном зале, на выбранном стеллаже и на выбранной полке. Результат должен быть выведен в Label4

К-22

Тип: ГЕНЕРАЦИЯ ПСЕВДОСЛУЧАЙНОГО МАССИВА (сортировка). ВХОДНЫЕ ДАННЫЕ формула, элементы массива ВЫХОДНЫЕ ДАННЫЕ ячейки листа

(Темы: объявление переменных; арифметические формулы и функции, массивы, циклы For ... Next, управление потоком выполнения программы IF ... THEN; доступ к ячейкам рабочего листа)

К-221

При помощи InputBox запросить количество элементов массива. Используя конструкцию Int((max-min+1)*Rnd+min), сгенерировать массив из n целочисленных элементов от 1 до n. Используя алгоритм сортировки, отсортировать массив и вывести результат в первый столбец Листа1

К-222

При помощи функции InputBox запросить количество элементов массива n. Используя конструкцию Int((max-min+1)*Rnd+min), сгенерировать массив из n+5 целочисленных элементов от 11 до n-1. Используя алгоритм сортировки, отсортировать полученный массив и вывести первоначальный массив в столбец B, а результат сортировки в третий столбец Листа1

К-23

Тип: ГЕНЕРАЦИЯ ПСЕВДОСЛУЧАЙНОГО МАССИВА (MIN, MAX элементы). ВХОДНЫЕ ДАННЫЕ формула,элементы массива ВЫХОДНЫЕ ДАННЫЕ ячейки листа, MsgBox

(Темы: объявление переменных; арифметические формулы и функции, массивы, циклы For ... Next, управление потоком выполнения программы IF ... THEN; доступ к ячейкам рабочего листа)

К-231

Используя конструкцию Int((max-min+1)*Rnd+min), сгенерировать массив из n целочисленных элементов от 0 до n, где n – число, записанное в ячейке на пересечении первого столбца и седьмой строки, находящейся на рабочем листе «Лист3». Вывести полученный массив в столбец B рабочего листа «Лист1», начиная со второй строки. Найти минимальный элемент массива, который должен быть выведен окно сообщений.

К-232

Необходимо, используя конструкцию Int((max-min+1)*Rnd+min), создать массив из k целочисленных элементов от 5 до k+5, где k – число, записанное в ячейке на пересечении второго столбца и девятнадцатой строки, находящейся на рабочем листе «Лист1». Вывести полученный массив в столбец E рабочего листа «Лист3», начиная со второй строки. Найти максимальный элемент массива, который должен быть выведен окно сообщений.

К-233

Используя конструкцию Int((max-min+1)*Rnd+min), сгенерировать массив из n целочисленных элементов от 0 до n, где n – число, записанное в ячейке на пересечении второго столбца и первой строки, находящейся на рабочем листе «Лист2». Вывести полученный массив в столбец C рабочего листа «Лист1», начиная со второй строки. Найти минимальный элемент массива, который должен быть выведен в ячейку A1 Листа1 и в окно сообщений.

К-234

Необходимо, используя конструкцию Int((max-min+1)*Rnd+min), создать массив из k целочисленных элементов от 3 до k+5, где k – число, записанное в ячейке на пересечении седьмого столбца и восьмой строки, находящейся на рабочем листе «Лист4». Вывести полученный массив в столбец D рабочего листа «Лист1», начиная со третьей строки. Найти максимальный элемент массива, который должен быть выведен окно сообщений.

К-24

Тип: ГЕНЕРАЦИЯ ПСЕВДОСЛУЧАЙНОГО МАССИВА (статистический анализ). ВХОДНЫЕ ДАННЫЕ формула, элементы массива ВЫХОДНЫЕ ДАННЫЕ ячейки листа, MsgBox

(Темы: объявление переменных; арифметические формулы и функции, массивы, циклы For ... Next, управление потоком выполнения программы IF ... THEN; доступ к ячейкам рабочего листа)

К-241

Используя конструкцию Int((max-min+1)*Rnd+min), сгенерировать массив из n целочисленных элементов от1 до n, где n – число, записанное в ячейке на пересечении первого столбца и седьмой строки, находящейся на рабочем листе «Лист5». Необходимо посчитать количество неповторяющихся (уникальных) элементов в данном массиве. Сам массив вывести в два столбца в окно сообщений. Результат подсчета должен быть выведен в последнюю строку того же окна сообщений

К-242

Воспользовавшись конструкцией Int((max-min+1)*Rnd+min), необходимо сгенерировать массив из n целочисленных элементов от 0 до n+5, где n – число, записанное в ячейке A5, находящейся на рабочем листе «Лист2». Необходимо посчитать количество повторяющихся (неуникальных) элементов в данном массиве. Сам массив вывести в три столбца в окно сообщений. Результат подсчета должен быть выведен в первую строку того же окна сообщений

К-243

Умело применив конструкцию Int((max-min+1)*Rnd+min), необходимо сгенерировать массив из n целочисленных элементов от 3 до n+5, где n – случайно полученное число больше ста и меньше 123. Необходимо вычислить несмещенную оценку мат.ожидания для данного массива. Сам массив вывести в четыре столбца в окно сообщений. Результат подсчета должен быть выведен в первую строку того же окна сообщений с текстовым комментарием

К-244

Творчески используя конструкцию Int((max-min+1)*Rnd+min), необходимо сгенерировать массив из z целочисленных элементов от 13 до n+7, где n – случайно полученное число больше 220 и меньше двухсот семидесяти пяти. Необходимо вычислить оценку среднеквадратического отклонения для элементов данного массива. Сам массив вывести в пять столбцов в окно сообщений. Результат подсчета должен быть выведен в последнюю строку того же окна сообщений с текстовым комментарием

К-245

Используя конструкцию Int((max-min+1)*Rnd+min), сгенерировать массив из n целочисленных элементов от1 до n, где n – число, записанное в ячейке на пересечении первого столбца и седьмой строки, находящейся на рабочем листе «Лист4». Необходимо посчитать количество неповторяющихся (уникальных) элементов в данном массиве. Сам массив вывести в три столбца в окно сообщений. Результат подсчета должен быть выведен в последнюю строку того же окна сообщений с текстовым комментарием

К-246

Воспользовавшись конструкцией Int((max-min+1)*Rnd+min), необходимо сгенерировать массив из P целочисленных элементов от 1 до P+6, где P – число, записанное в ячейке A9, находящейся на рабочем листе «Лист1». Необходимо посчитать количество повторяющихся (неуникальных) элементов в данном массиве. Сам массив вывести в три столбца в окно сообщений. Результат подсчета должен быть выведен в первую строку того же окна сообщений

K-247

Умело применив конструкцию Int((max-min+1)*Rnd+min), необходимо сгенерировать массив из n целочисленных элементов от 2 до n+3, где n – любое число больше ста двадцати и меньше 122. Необходимо вычислить несмещенную оценку мат.ожидания для данного массива. Сам массив вывести в четыре столбца в окно сообщений. Результат подсчета должен быть выведен в первую строку того же окна сообщений с текстовым комментарием

K-248

Творчески используя конструкцию Int((max-min+1)*Rnd+min), необходимо сгенерировать массив из z целочисленных элементов от 6 до n+17, где n – случайно полученное число больше 220 и меньше двухсот семидесяти трех. Необходимо вычислить оценку среднеквадратического отклонения для элементов данного массива. Сам массив вывести в шесть столбцов в окно сообщений. Результат подсчета должен быть выведен в последнюю строку того же окна сообщений с текстовым комментарием

K-25

Тип: НАПИСАНИЕ ПРОГРАММЫ ПО БЛОК-СХЕМЕ АЛГОРИТМА ВХОДНЫЕ ДАННЫЕ блок-схема ВЫХОДНЫЕ ДАННЫЕ программа

(Темы: объявление переменных; арифметические формулы, циклы For ... Next, управление потоком выполнения программы IF ... THEN; доступ к ячейкам рабочего листа)

К-251

На языке VBA написать программу, реализующую алгоритм, заданный блок-схемой, изображенной на Рис.21.1

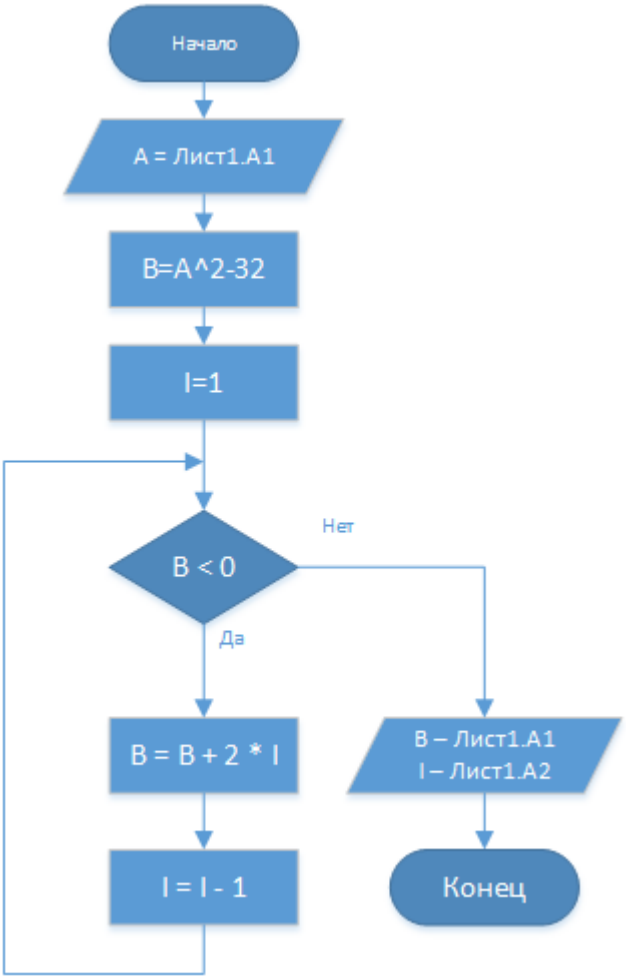

Рис.21.1

К-252

На языке VBA написать программу, реализующую алгоритм, заданный блок-схемой, изображенной на Рис.21.2

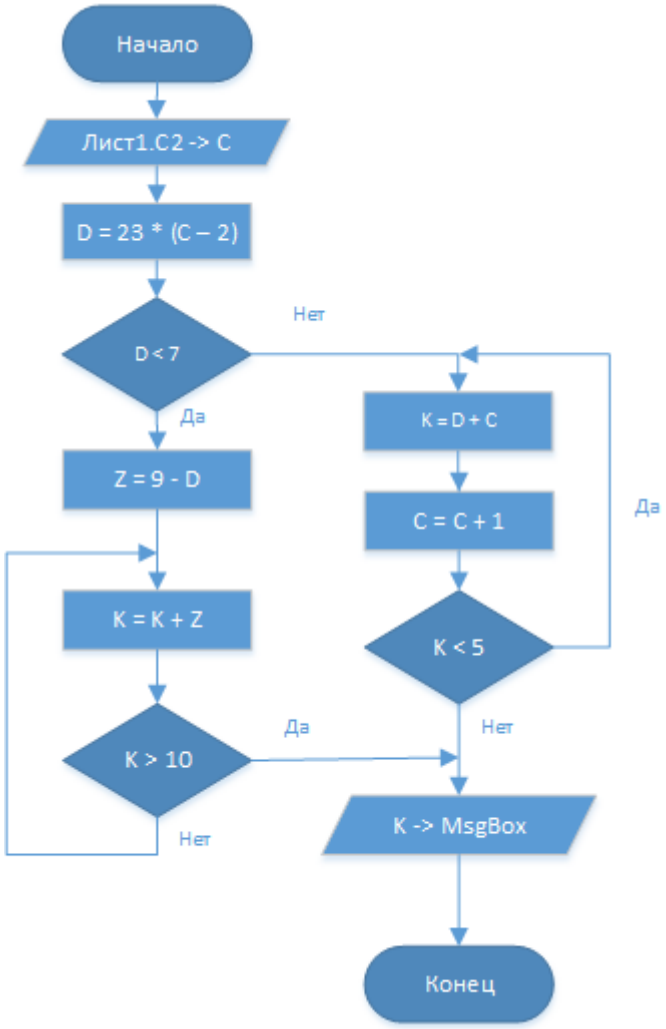

Рис.21.2

22. Приложение 3. Проектные задания

22.1. Написание библиотеки матричных операций

При решении задач моделирования экономических и производственных процессов во многих случаях встречаются вычисления связанные с матричными операциями. Поэтому умение писать программы, автоматизирующие выполнение таких операций может послужить полезным опытом для студентов. В рамках данного проекта можно поставить задачу максимум – написать программу для решения линейных уравнений произвольного порядка (не выше заранее определенного). Далее, для простоты реализации можно основную задачу разбить на подзадачи и попросить студентов написать процедуры и функции, выполняющие следующие математические операции:

- нахождение минора
- нахождение определителя
- нахождение обратной матрицы
- нахождение произведения вектора на вектор
- нахождение произведения матрицы на вектор
- нахождение суммы матриц одинаковой размерности
- нахождение суммы векторов одинаковой размерности

Ниже в качестве примера приведена возможная программная реализация решения перечисленных выше задач. Отметим, что для простоты реализации размерность участвующих в выполнении операции операндов передается в процедуры и функции в качестве параметров, а сами матрицы и векторы, как входные, так и результирующие передаются по ссылке.

```
 Sub sum_matr(ByRef a() As Double, ByRef b() As Dou-
ble, ByRef mo() As Double, x As Integer, y As Inte-
ger)
```

'суммирование матриц a и b, результат в c (x строк y столбцов)

```
  For i = 0 To x - 1
    For j = 0 To y - 1
      mo(i) = a(i) + b(i)
    Next j
  Next i
End Sub

Sub sum_vect(ByRef vi1() As Double, ByRef vi2() As
Double, ByRef vo() As Double, x As Integer)
```

' суммирование векторов vi1 и vi2, результат в vo (x разрядов)

```
  For i = 0 To x - 1
       vo(i) = vi1(i) + vi2(i)
  Next i
End Sub

Sub matr_x_vect(ByRef a() As Double, ByRef vi() As
Double, ByRef vo() As Double, x As Integer, y As In-
teger)
```

'произведение матрицы a на вектор vi, результат в vo ' (размерность матрицы: x строк, y столбцов, вектора: y разрядов)

```
  For i = 0 To x - 1
    vo(i) = 0
    For j = 0 To y - 1
      vo(i) = vo(i) + a(i, j) * vi(j)
    Next j
  Next i
End Sub

Function vect_x_vect(ByRef vi1 As Double, vi2 As Dou-
ble, z As Integer) As Double
```

' функция возвращает реультат скалярного произведения векторов vi1 и vi2, размерностью z разрядов

```
  vect_x_vect = 0
  For i = 0 To z - 1
    vect_x_vect = vect_x_vect + vi1(i) * vi2(i)
  Next i
End Function
Function minor2(ByRef a() As Double) As Double
```

' функция вычисляет определитель *2x2*

```
 minor2 = a(0, 0) * a(1, 1) - a(0, 1) * a(1, 0)
End Function
Sub tominor(ByRef a() As Double, ByRef b() As Double,
         ii As Integer, jj As Integer, zz As Integer)
```

'процедура получает минор для ii, jj элемента матрицы a и
записывает в матрицу b

```
 For i = 0 To zz
   For j = 0 To zz
   If i < ii And j < jj Then
       b(i, j) = a(i, j)
     Else
       If i < ii And j > jj Then
```

```
          b(i, j - 1) = a(i, j)
        Else
          If i > ii And j < jj Then
            b(i - 1, j) = a(i, j)
          Else
            If i > ii And j > jj Then
               b(i - 1, j - 1) = a(i, j)
            End If
          End If
         End If
        End If

   Next j
 Next i
End Sub

Sub show_matr(ByRef a() As Double, x As Integer, y As
                        Integer, Optional st As String)
```

' отображает матрицу a размерности *x, y в окне MsgBox*

```
  Dim ss As String
  For i = 0 To x - 1
    For j = 0 To y - 1
      ss = ss & Round(a(i, j), 3) & vbTab
    Next j
    ss = ss & vbLf
  Next i
  MsgBox ss, , st
End Sub

Sub show_vect(ByRef v() As Double, x As Integer,
                               Optional st As String)
```

'отображает вектор v (x разрядов)

```
  Dim ss As String
  For i = 0 To x - 1
    ss = ss & Round(v(i), 3) & vbTab
  Next i
  MsgBox ss, , st
End Sub

Function minorx(ByRef a() As Double, ii As Integer)
As Double
```

'вычисляет определитель квадратной матрицы a размерности ii x ii

```
Dim m(10, 10) As Double
Dim i As Integer
Dim ss As Integer
Dim mm As Integer
ss = 0
For i = 0 To ii - 1
 Call tominor(a, m, 0, i, ii - 1)
 If ii = 3 Then
   mm = minor2(m) * (-1) ^ (i Mod 2)
 Else
   mm = minorx(m, ii - 1) * (-1) ^ (i Mod 2)
 End If
 ss = ss + a(0, i) * mm
Next i
minorx = ss
End Function

Sub transp_matr(ByRef a() As Double, ByRef b() As
Double, xx As Integer)
```

'транспонирует квадратную матрицу a, результат в b (размерность xx)

```
  For i = 0 To xx - 1
    For j = 0 To xx - 1
      b(i, j) = a(j, i)
    Next j
  Next i
End Sub

Sub inv_matr(ByRef a() As Double, ByRef b() As Dou-
ble, xx As Integer)
```

‘находит матрицу b обратную квадратной матрице a размерности xx

```
Dim c(10, 10) As Double
Dim bb(10, 10) As Double
Dim i As Integer, j As Integer
Dim det_x As Double

det_x = minorx(a, xx)
If det_x = 0 Then
  MsgBox "det=0"
  Exit Sub
End If

For i = 0 To xx - 1
  For j = 0 To xx - 1
```

```
    Call tominor(a, c, i, j, xx - 1)
    If xx = 3 Then
      bb(i, j) = minor2(c) * (-1) ^ (i + j)
    Else
      bb(i, j) = minorx(c, xx - 1) * (-1) ^ (i + j)
    End If
  Next j
Next i

Call transp_matr(bb, b, xx)
For i = 0 To xx - 1
  For j = 0 To xx - 1
    b(i, j) = b(i, j) / det_x
  Next j
Next i
End Sub

Sub mmz()
```

'процедура, показывающее «создание» и решение систем линейных уравнений

```
Dim i As Integer
Dim a() As Variant
Dim b() As Variant
Dim z() As Variant
Dim c(3, 3) As Double
Dim d(4, 4) As Double
Dim e(5, 5) As Double

Dim bb(10, 10) As Double
Dim vz(10) As Double

Dim vv3 As Variant
Dim vx3(3) As Double
Dim vb3(3) As Double

Dim vv4 As Variant
Dim vx4(4) As Double
Dim vb4(4) As Double

Dim vv5 As Variant
Dim vx5(5) As Double
Dim vb5(5) As Double
```

```
vv3 = [{1,2,3}]                 ' инициализация вектора
                                  размерности 3
vv4 = [{1,2,3,4}]               ' инициализация вектора
                                  размерности 4
vv5 = [{1,2,3,4,5}]             ' инициализация вектора
                                  размерности 5
a = [{1,2,3;4,1,6;7,8,9}]       ' инициализация матрицы
3x3
b = [{1,2,3,4;4,1,6,7;7,8,9,10;11,11,12,13}]
' инициализация матрицы 4x4
z = [{1,2,3,4,5;4,1,6,7,8;7,8,9,10,11;11,11,12,13,14;
                                    15,16,17,18,18}]
' инициализация матрицы 5x5
' При инициализации нумерация элементов начинается с 1, а нам надо с 0
' Новыми  матрицами будут  матрицы c, d и e

For i = 0 To 2
 vx3(i) = vv3(i + 1)
 For j = 0 To 2
   c(i, j) = a(i + 1, j + 1)
 Next j
Next i

For i = 0 To 3
 vx4(i) = vv4(i + 1)
 For j = 0 To 3
   d(i, j) = b(i + 1, j + 1)
 Next j
Next i

For i = 0 To 4
 vx5(i) = vv5(i + 1)
 For j = 0 To 4
   e(i, j) = z(i + 1, j + 1)
 Next j
Next i
```

'Находим векторы vb для составления линейных уравнений A*x=vb

```
Call matr_x_vect(c, vx3, vb3, 3, 3)
Call show_vect(vb3, 3, "Вектор b3")
Call matr_x_vect(d, vx4, vb4, 4, 4)
Call show_vect(vb4, 4, "Вектор b4")
Call matr_x_vect(e, vx5, vb5, 5, 5)
```

```
Call show_vect(vb5, 5, "Вектор b5")
```

'Находим обратную матрицу для матрицы c

```
Call inv_matr(c, bb, 3)
Call show_matr(bb, 3, 3, "Матрица обратная c")
```

'Находим решение уравнения

```
Call matr_x_vect(bb, vb3, vz, 3, 3)
Call show_vect(vz, 3, "Решение уравнения")
```

'Находим обратную матрицу для матрицы d

```
Call inv_matr(d, bb, 4)
Call show_matr(bb, 4, 4, "Матрица обратная d")
```

'Находим решение уравнения

```
Call matr_x_vect(bb, vb4, vz, 4, 4)
Call show_vect(vz, 4, "Решение уравнения")
```

'Находим обратную матрицу для матрицы c

```
Call inv_matr(e, bb, 5)
Call show_matr(bb, 5, 5, "Матрица обратная e")
```

'Находим решение уравнения

```
Call matr_x_vect(bb, vb5, vz, 5, 5)
Call show_vect(vz, 5, "Решение уравнения")

End Sub
```

22.2. Написание программы аутентификации пользователя

Программная реализация задачи аутентификации пользователя предполагает решение следующих подзадач:

- задание пользовательского пароля;
- сохранение пароля в защищенном виде;
- проверка на соответствие логина и пароля.

В свою очередь, одно из решений подзадачи сохранения пароля в защищенном виде предполагает преобразование пароля при помощи односторонней функции, которую мы назвали квазихэш функцией из-за ее простоты и неполного соответствия требованиям, предъявляемым к реальным хэшфункциям. В качестве односторонней функции мы использовали операцию нахождения остатака по модулю какого-то простого числа a от 30 до 110 (операция mod a) для кода каждого из символов пароля. Для наглядности при разборе проекта рекомендуется вместе со студентами разработать укрупненные блок-схемы алгоритмов для каждой из описанных выше операций.

Форма для выполнения описанных выше операций может выглядеть таким образом, как это показано на рисунке Рис 22.1.

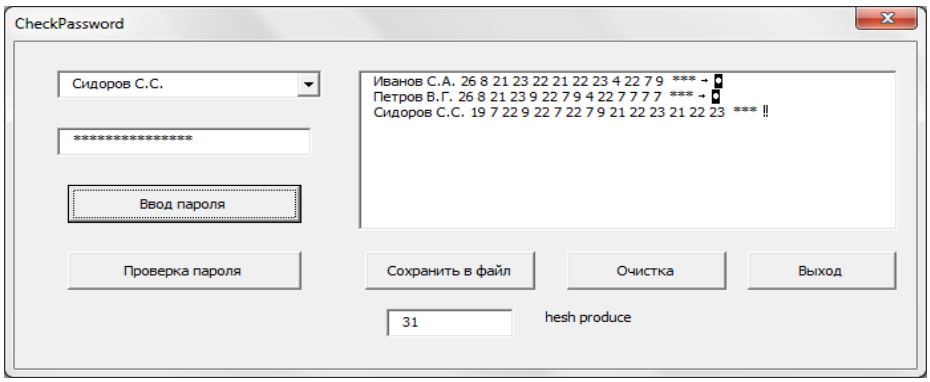

Рис. 22.1

В качестве пользователей рассмотрены для наглядности только три пользователя: Иванов, Петров и Сидоров, которые заносятся в элемент управления ComboBox во время инициализации формы. Опять же для наглядности после очередного изменения пароля логин и хэш пароля должны отображаться в элементе управления ListBox. Для упрощения выполнения файловых операций чтения /записи запись хэшей паролей в файл разрешена только после того, как будут заполнены все пароли.

Операция проверки соответствия логина и пароля проводится в несколько этапов. Файл с паролями читается построчно и производится поиск подстроки содержащей идентификатор выбранный в ComboBox. При совпадении идентификаторов выполняется проверка длины проверяемого пароля и длины хэш для данного логина, записанного в файл. При совпадении и этих

параметров проводится посимвольное сравнение. Для придания реалистичности разрабатываемому проекту можно поставить дополнительную задачу ведения протоколирования в текстовый файл всех попыток (и удачных и не удачны) проверки пароля.

Ниже показан пример кода программы, реализующей данных проект. Отметим, что вся программа реализована в виде функций пользовательской формы.

```
Dim pw(3, 20) As Integer
Dim Names(3) As String
Dim try_pwd(200) As String
Dim try_pwd_ind As String
Dim hesh_x As Integer

Private Sub set_pw(u_num As Integer)
```

' *** установка пароля ***

```
If Len(TextBox2.Text) = 0 Then
  MsgBox "Введите Хэш"
  Exit Sub
End If
hesh_x = TextBox2.Value
StrBuf = TextBox1.Text
sbL = Len(StrBuf)
  If sbL < 3 Then
    MsgBox "Плохой пароль!" & vbLf & "Повторите ввод"
    Exit Sub
  End If
pw(u_num, 0) = sbL
For i = 0 To sbL - 1
  pw(u_num, i + 1) = Asc(Mid(StrBuf, i + 1, 1)) Mod
hesh_x
Next i
Call pw_show
End Sub


Private Sub check_pw(u_num As Integer, f_name As
String)
```

' *** проверка пароля ***

```
 Dim u_name As String
 Dim u_pw As String
 Dim u_pw_hesh(20) As Integer
 Dim StrBuf As String
```

```
 Dim path_x As String
 Dim hFileIn As Long
 Dim buf_info() As String
 Dim a As Integer, b As Integer
 Dim str_log As String

 path_x = ThisWorkbook.Path & "\"
 u_name = ComboBox1.Value
 str_log = u_name + "  "
 u_pw = TextBox1.Text
 If Len(u_pw) <= 3 Then
```

'*** Введен неверный пароль ***

```
   str_log = str_log & " Длинна пароля <=3 " & Now()
   try_pwd(try_pwd_ind) = str_log
   try_pwd_ind = try_pwd_ind + 1
   Call try_pwd_into_list
   Exit Sub
 End If
 StrBuf = Dir(path_x + f_name)
 If Len(StrBuf) = 0 Then
```

'*** Нет файла паролей ***

```
   str_log = str_log & " Нет файла паролей " & Now()
   try_pwd(try_pwd_ind) = str_log
   try_pwd_ind = try_pwd_ind + 1
   Call try_pwd_into_list
   Exit Sub
 End If
 hFileIn = FreeFile
 Open path_x + f_name For Input Access Read As
#hFileIn
 Do Until EOF(hFileIn)
  Line Input #hFileIn, StrBuf
  p = InStr(1, StrBuf, u_name)
  If p > 0 Then
   buf_info = Split(Trim(StrBuf), " ")
   If Len(u_pw) <> UBound(buf_info) - 1 Then
```

'*** уточнить пароль ***

```
     str_log = str_log & " Пароль не совпал по длинне
" & Now()
      try_pwd(try_pwd_ind) = str_log
      try_pwd_ind = try_pwd_ind + 1
      Call try_pwd_into_list
```

```
      Close hFileIn
      Exit Sub
    End If
    For i = 2 To UBound(buf_info)
     a = Asc(Mid(u_pw, i - 1, 1)) Mod hesh_x
     b = buf_info(i)
     If a <> b Then
```

'*** уточнить пароль ***

```
       str_log = str_log & " Пароль не совпал по сим-
волам " & Now()
       try_pwd(try_pwd_ind) = str_log
       try_pwd_ind = try_pwd_ind + 1
       Call try_pwd_into_list
       Close hFileIn
       Exit Sub
     End If
    Next i
    MsgBox "All OK"
```

'*** Успешный пароль ***

```
    str_log = str_log & " Успешный пароль " & Now()
    try_pwd(try_pwd_ind) = str_log
    try_pwd_ind = try_pwd_ind + 1
    Call try_pwd_into_list
    Exit Do
   End If
 Loop
 Close hFileIn
End Sub


Private Sub try_pwd_into_list()
  ListBox1.Clear
  For i = 0 To try_pwd_ind
    ListBox1.AddItem try_pwd(i)
 Next
End Sub

Private Sub CommandButton1_Click()
```

'*** set password ***

```
Dim sbL As Integer
Dim StrBuf As String
```

```
Dim numP As Integer
numP = ComboBox1.ListIndex
Call set_pw(numP)

End Sub

Private Sub CommandButton2_Click()
 '*** Check password ***
  Call check_pw(ComboBox1.ListIndex, "pwd.txt")

End Sub

Private Sub CommandButton3_Click()
'*** exit ***
  Unload UserForm2
End Sub

Private Sub CommandButton4_Click()
 '*** PW into FILE ***
  Call file_pw("pwd.txt")
End Sub

Private Sub UserForm_Initialize()
 TextBox1.PasswordChar = "*"
 hesh_x = 31
 TextBox2.Value = hesh_x
 Names(0) = "Иванов С.А."
 Names(1) = "Петров В.Г."
 Names(2) = "Сидоров С.С."
 For i = 0 To 2
   ComboBox1.AddItem Names(i)
 Next
 ComboBox1.ListIndex = 0
 For i = 0 To 200
  try_pwd(i) = ""
 Next
 try_pwd_ind = 0
End Sub
Sub pw_show()
Dim strbuf1 As String
Dim i As Integer
ListBox1.Clear
```

```
For i = 0 To 2
  strbuf1 = Names(i) + " "
  For j = 1 To pw(i, 0)
    strbuf1 = strbuf1 & pw(i, j) & " "
  Next j
  strbuf1 = strbuf1 & " *** " & pw_preobr(i)
  ListBox1.AddItem strbuf1
Next i
End Sub

Function pw_preobr(ind As Integer) As String

Dim sbuf As String
For i = 1 To pw(ind, 0)
   sbuf = sbuf & Chr(pw(ind, i))
Next
pw_preobr = sbuf
End Function

Function is_hesh() As Boolean
For i = 0 To 2
  If pw(i, 0) = 0 Then
    is_hesh = False
    Exit Function
  End If
Next i
is_hesh = True
End Function

Private Sub file_pw(fname As String)
  Dim hFile As Long
  Dim path_x As String
  Dim StrBuf As String

  path_x = ThisWorkbook.Path + "\"
  If Not is_hesh Then
    MsgBox "Все пароли должны быть введены"
    Exit Sub
  End If
  hFile = FreeFile
  Open path_x + fname For Output Access Write As
#hFile
  For i = 0 To 2
```

```
    StrBuf = Names(i) & " "
    For j = 1 To pw(i, 0)
      StrBuf = StrBuf & pw(i, j) & " "
    Next j
    Print #hFile, Trim(StrBuf)
  Next i
  Close #hFile
  MsgBox "Пароли сохранены успешно"
End Sub
```

23. Литература

1. Михеев Р. VBA и программирование в MS Office для пользователей / Михеев Р. – СПб.:БХВ-Петербург, 2006.– 369 с.

2. Гарнаев А. Использование MS Excel и VBA в экономике и финансах / А. Гарнаев. – М: ОЗОН-Пресс, 2012. – 336 с.

3. Гарнаев А. Excel, VBA, Internet в экономике и финансах / А. Гарнаев. – СПб: БХВ-Петербург, 2005. – 816 с.

4. Кузьменко В. VBA. Эффективное использование / В.Кузьменко. – М.: Бином-Пресс, 2012. – 624 с.

5. Карлберг К. Бизнес-анализ с использованием Excel (Business Analysis: Microsoft Excel 2010) / Карл Карлберг; пер. с англ. В. Гинзбург. – М.: Вильямс, 2014. – 576 с.

6. Шеферд Р. Как облегчить себе жизнь и увеличить производительность в Microsoft Excel с помощью макросов / Р. Шеферд; пер. с англ. Р.М. Ефтеева. – М.: НТ Пресс, 2007. – 352 с.: ил.

7. Гетц К. Программирование в Microsoft Office. Для пользователя / К. Гетц, М. Джилберт; пер. с англ. – К.: Издательская группа BHV, 2000. – 384 с.

8. Гетц К. Программирование в Microsoft Office. Полное руководство по VBA / К. Гетц, М. Джилберт; пер. с англ. – К.: Издательская группа BHV, 2000. – 768 с.

9. Биллиг В.А. Средства разработки VBA-программиста. Офисное программирование. Том 1. / В.А. Биллиг. – М.: Изательско-торговый дом «Русская редакция», 2001. – 480 с.: ил.

10. Создание форм и самые важные свойства и методы форм [Электронный ресурс]. – Режим доступа: http: // www.askit.ru / custom / vba_office / m5 / 05_01_forms_basics.htm

11. Бейсик. QBasic и Visual Basic / Я.Н. Глинский, В.Е. Анохин, В.А. Ряжская.- СПб.: ООО «ДиаСофтЮП», 2002. – 192 с.

12. Компьютерное моделирование экономики / И.Ф. Цисарь, В.Г. Немейман – М.: «Диалог-МИФИ», 2002. – 304 с.