Министерство образования и науки российской федерации
Северный (Арктический) федеральный университет
им. М. В. Ломоносова

А. С. Грошев

Программирование на языке Microsoft Visual Basic for Applications

Методические указания
к выполнению лабораторных работ

Архангельск
2013

Рассмотрено и рекомендовано к изданию методической комиссией
__ _____ 2013 г.

Рецензент

_____

_____

УДК 004.65
ББК 32.973.26 - 018.2
Г 89
Грошев А.С. Программирование на языке Microsoft Visual Basic for Applications: Метод. указания к выполнению лабораторных работ. – Архангельск, Изд-во Арханг. гос. техн. ун-та, 2013. – 35 с.

Рассмотрены основы программирования на алгоритмическом языке Visual Basic for Application в системах Microsoft Word, Excel и Access.

Приведены примеры программ с описанием и комментариями в их тексте.

Предназначено для студентов вузов.

Ил. 19. Табл. 7. Библиогр. 3 назв.

ОГЛАВЛЕНИЕ

Лабораторная работа №1. Программирование на языке VBA в Microsoft Office Word

Microsoft Visual Basic for Applications (VBA) является встроенным языком программирования для приложений Microsoft Office (Word, Excel, Access, PowerPoint, Outlook, FrontPage, InfoPath), а также и для некоторых других системах (Microsoft Visio и Project, CorelDRAW, CorelWordPerfect Office 2000, AutoCAD). Одновременно существует новый способ разработки приложений Office – использование средств Visual Studio Tools for Office (VSTO).

VBA имеет очень много общего с VBScript [1].

Наиболее существенные отличия – в правилах написания идентификаторов, в описании переменных, соглашении об именовании процедур:

1) Идентификаторы в языке VBA в русской версии Microsoft Office могут использовать русские буквы. Нельзя использовать пробел, точку, символы !, @, &, $, #. Первый символ в имени – обязательно буква.
2) Переменные и массивы могут быть объявлены с указанием их типа **Dim <имя> [As <тип>]**. Типы переменных см. [1, таблица 5.4].
3) Процедуры могут быть общими и событийными (процедуры обработки событий для объектов). Имя событийных процедур состоит из имени объекта и имени события, между которыми стоит символ подчеркивания: **Sub <ИмяОбъекта_ИмяСобытия>**, например **Sub Кнопка1_Click()**.

При программировании на языке VBA используются библиотеки объектных типов (для Word это файл *MSWORD.OLB*). Информацию о типах объектов можно найти в Обозревателе объектов (Object Browser) в редакторе Microsoft Visual Basic, встроенном в систему Word и другие приложения Office (в меню пункт View – Object Browser).

На странице документа Word можно организовать работу с достаточно сложной программой, даже не создавая для этого отдельных Windows-форм, запускающихся, например, при открытии документа или при нажатии на кнопку на странице документа, а просто разместив на странице поля ввода данных и показывая программно на той же странице результаты после изменения данных.

Однако, многие математические расчеты проще выполнить в системе Excel, при хранении информации в базах данных достаточно большие возможности обработки этой информации имеет Access. При работе с базами данных в крупных информационных системах может быть предусмотрено формирование отчетов с выводом их в Word или

Excel, при этом разработку таких программ может выполнить лишь специалист, хорошо знакомый как с архитектурой базы данных информационной системы, так и с языком VBA.

Некоторые принципы работы с объектами Word можно освоить, если записать некоторую последовательность своих действий с помощью средства Word ***Запись макроса***, которое присутствует на вкладке ленты ***Разработчик***. Данная вкладка присутствует, если в ***Параметрах Word*** на странице ***Основные*** стоит галочка у пункта ***Показывать вкладку Разработчик на ленте*** (в старых версиях) или в ***Параметрах Word*** в разделе ***Настроить ленту*** стоит галочка для вкладки ***Разработчик*** (Word 2010, 2013).

Макрос – текст программы, сохраненный в формате документа Office с поддержкой макросов. В настоящее время эта программа является процедурой, написанной на языке **VBA.**

Выполним следующие действия:

1) нажмем на кнопку ***Запись макроса***, ни кнопку, ни клавиши макросу можно не назначать, зададим, где будет сохраняться текст макроса: ***Макрос доступен для*** текущего документа;
2) переместимся в конец документа с текущей позиции (была не в конце), нажав клавиши *Ctrl+End*;
3) нажмем клавишу *Enter* и напишем слово «Привет!»;
4) еще раз повторим операцию пункта 3;
5) выделим последнее слово, нажав Shift+Home и нажмем клавишу *Delete*;
6) перейдем в начало первой строки «Привет!», нажмем Shift+End и затем клавишу *Delete*;
7) на вкладке разработчик сначала нажмем кнопку ***Остановить запись***, затем кнопку ***Visual Basic***;
8) в открывшемся окне редактора **Microsoft Visual Basic** слева на панели **Project** сделаем двойной щелчок мышкой на разделе ***Progect*(<текущий файл>) – *Modules* – *NewMacros*** и увидим справа окно с текстом программы на языке VBA, которое показано на рисунке 1.1.

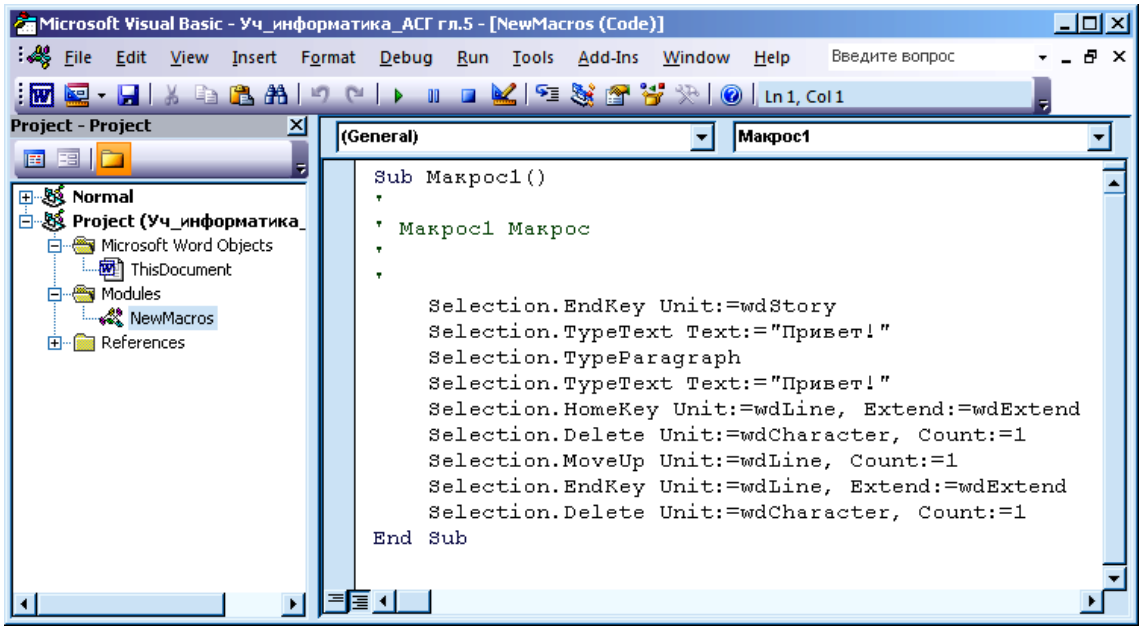


Рисунок 1.1 – Текст макроса на языке VBA в редакторе **Microsoft Visual Basic**

При сохранении документа, в котором присутствуют программные модули, следует выбрать тип файла ***Документ Word с поддержкой макросов***, иначе текст программ не будет сохранен.

В тексте Макроса1 (Sub Макрос1() – процедупа языка VBA, см. рисунок 1.1) используется объект ***Selection*** – место, где находится курсор в текущем документе (объект ***Application***) или выделенный фрагмент документа. Полный синтаксис обращения к этому объекту: ***Application.Selection***…, но имя родительского объекта можно опустить, так как при работе с документом в системе Word объект ***Application*** всегда является текущим активным объектом.

Текст Макроса1 с комментариями:

```
Selection.EndKey Unit:=wdStory
        ‘ переместить курсор в конец всего документа
Selection.TypeText Text:="Привет!"
        ‘ написать текст
Selection.TypeParagraph
        ‘ перейти к новому абзацу
Selection.TypeText Text:="Привет!"
        ‘ написать текст
Selection.HomeKey Unit:=wdLine, Extend:=wdExtend
‘ выделить фрагмент текста от текущей позиции до начала строки
Selection.Delete Unit:=wdCharacter, Count:=1
        ‘ удалить выделенный фрагмент
Selection.MoveUp Unit:=wdLine, Count:=1
```

‘ переместить курсор вверх на одну строку

Selection.EndKey Unit:=wdLine, Extend:=wdExtend

‘ выделить строку

Selection.Delete Unit:=wdCharacter, Count:=1

‘ удалить строку

Выполнить макрос можно, если на вкладке ленты Word **Разработчик** выбрать команду **Макросы**, затем в списке выбрать Макрос1 и нажать кнопку **Выполнить**. Две строки текста очень быстро появятся и исчезнут.

Работу макросов можно проследить с использованием отладочных режимов в окне **Visual Basic** (рисунок 1.2).

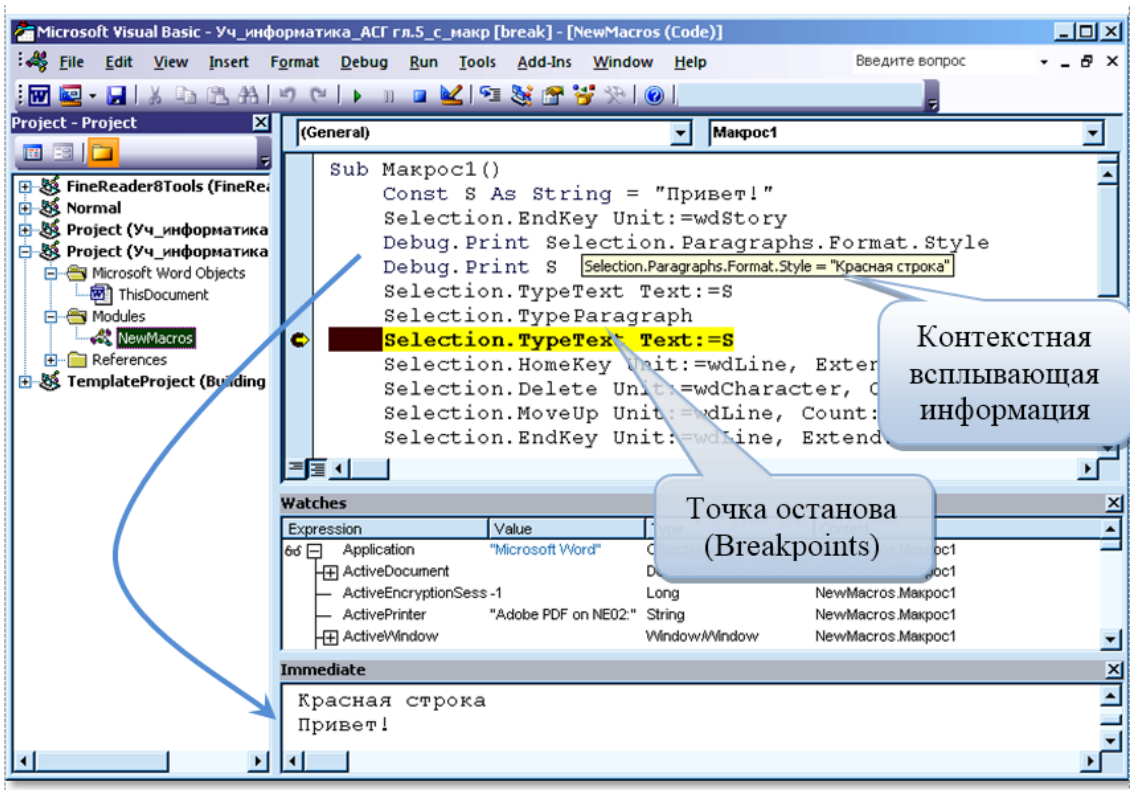


Рисунок 1.2 – Отладочные режимы в окне Microsoft Visual Basic

В пункте меню **Debug** (рисунок 1.3) и на панели инструментов присутствуют команды:

Step Into – выполнить построчно с заходом из текущей процедуры во все вызываемые процедуры;

Step Over – выполнить построчно без захода из текущей процедуры в вызываемые процедуры;

Step Out – выполнить остающиеся строки с заходом из текущей процедуры во все вызываемые процедуры;

Run To Cursor – выполнить до позиции курсора в текущей процедуре;

Add Watch – добавить объект или переменную в окно просмотра **Watches**.

Toggle Breakpoints – задать точку останова в программе. Точку останова можно также задать щелчком мыши на левой вертикальной рамке окна программы.

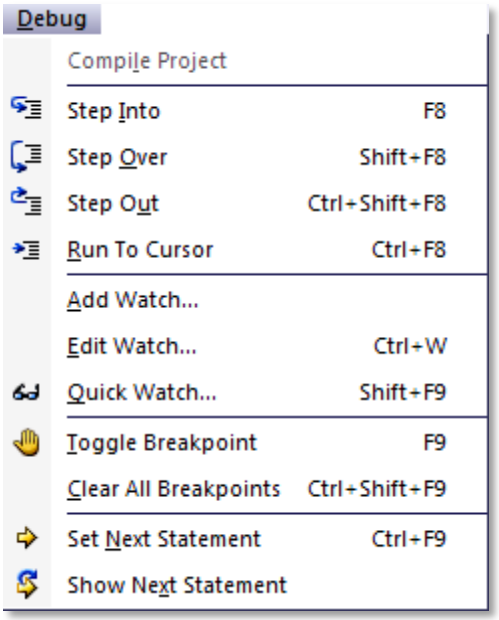


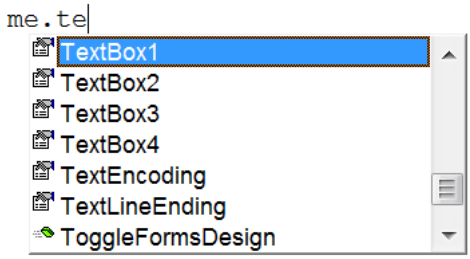
Рисунок 1.3 – Пункты меню Debug системы VBA

Так, задав точку останова на четвертой строке Макроса1 и запустив макрос, после остановки выполнения программы на этой строке в окне ***Watches*** можно просмотреть все свойства текущего документа (см. рисунок 1.2). Во время останова можно просмотреть значения свойств и переменных во всплывающих подсказках, появляющихся после наведения курсора мыши на имя свойства или переменной. Далее программу можно выполнить по одной строке и следить за ее работой. Текст Макроса1 был немного дополнен: добавлена константа S и задан вывод значений командой ***Debug.Print*** (вывод идет в отладочное окно ***Immediate***).

Синтаксические ошибки в тексте программы автоматически выделяются красным цветом с показом окна сообщения об ошибке. Если в меню окна Visual Basic в пункте *Tools-Options* убрать галочку у строки *Auto Syntax Check*, выделение ошибок красным цветом останется, но окно сообщений об ошибке появляться не будет.

При написании текста программ можно использовать всплывающий список свойств и методов, который появляется для стандартных объектов в редакторе **Visual Basic** после написания имени объекта и нажатия после неё точки (рисунок 1.4). В примере зарезервированное слово **Me** – текущий объект, в показанном случае обращение к свойству **Me.TextBox1.Value** то же самое, что и **ThisDocument.TextBox1.Value**.

Всплывающий список свойств и методов можно получить также, нажав комбинацию клавиш Ctrl+пробел.



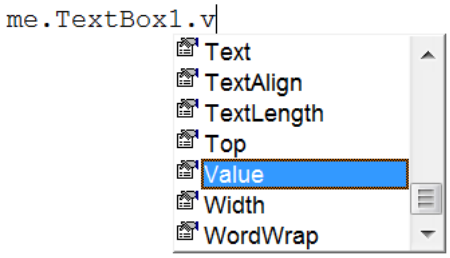


Рисунок 1.4 – Использование всплывающих списков свойств и методов объекта (появляются после нажатия точки после имени объекта)

Кроме того, при написании текста программы для стандартный процедур и функций появляются всплывающие подсказки по их синтаксису (см. рисунок 1.5).

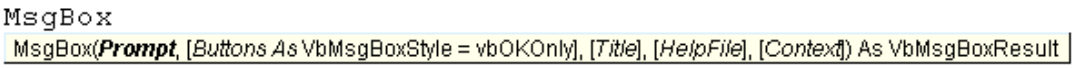


Рисунок 1.5 – Всплывающая подсказка в редакторе VBA

На странице документа могут быть размещены стандартные объекты Word и объекты ActiveX, присутствующие на вкладке **Разработчик** – **Элементы управления** (показаны на рисунке 1.6).

Рисунок 1.6 – Элементы управления для документа Word

Простой пример организации вычислений на странице документа – три объекта типа **Поле:**

11,1237 + 12,3456 = 23.4693

Двойной щелчок на поле в **Режиме конструктора** (в котором мы оказываемся, когда выбираем операцию *Добавить поле* из **Элементов управления**) перемещает в редактор Visual Basic, в нем создается пустая событийная процедура ***Private Sub TextBox1_Change()***, аналогично для второго поля (событие Change – изменение объекта, точнее

в данном случае изменение текста в объекте типа Поле). Напишем в этих процедурах одну строку – обращение к процедуре ***Calc1*** и создадим общую процедуру расчета ***Private Sub Calc1().*** Текст процедур будет выглядеть следующим образом:

```
Private Sub TextBox1_Change()
        ' процедура для события Change объекта TextBox1
    Calc1
End Sub
```

```
Private Sub TextBox2_Change()
        ' процедура для события Change объекта TextBox1
    Calc1
End Sub
```

```
Private Sub Calc1()
  If Me.TextBox1 = "" Or Me.TextBox2 = "" Then
   Me.TextBox3 = ""
   Exit Sub
  End If
  On Error Resume Next
  Me.TextBox3 = CSng(Me.TextBox1) +
CSng(Me.TextBox2)
  If Err <> 0 Then
     MsgBox "Задайте число в формате, " & _
     "определенном в установках Windows", _
     vbCritical, "Ошибка ввода"
   End If
 End Sub
```

Для третьего поля в окне **Properties** зададим свойство Enabled равным False, как показано на рисунке 1.7.

В результате после написания или изменения чисел в объектах ***TextBox1*** и ***TextBox2*** в поле ***TextBox3*** будет отображаться математическая сумма двух чисел.

Если в процедуре ***Calc1*** написать расчет без преобразования данных в числовое значение (***CSng(TextBox1)*** или ***CSng(TextBox1.Value)***), т. е. ***TextBox3 = TextBox1 + TextBox2***, то вместо математической суммы получится сложение двух строковых значений.

Другой пример – использование объекта типа **Кнопка** для расчета количества таблиц в текущем документе:

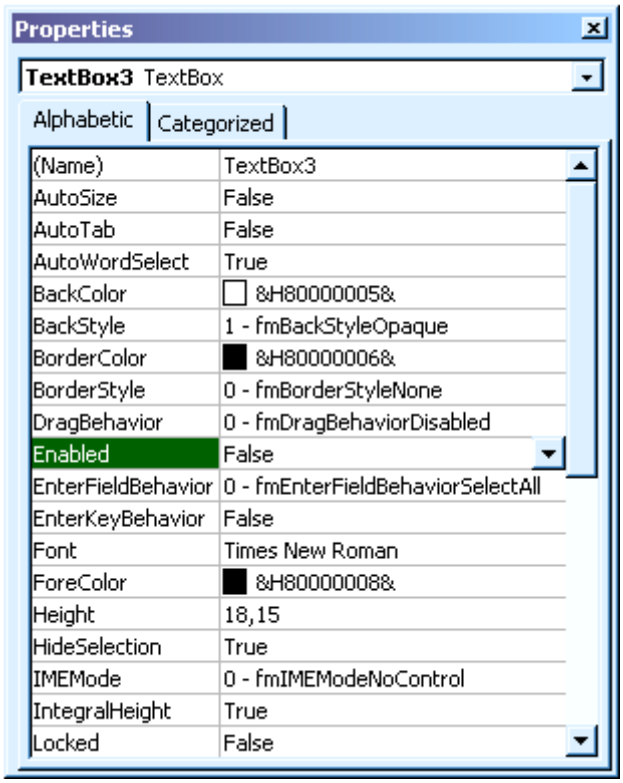


Рисунок 1.7 – Окно свойств объекта

С объектом ***CommandButton1*** связана следующая событийная процедура:

```
Private Sub CommandButton1_Click()
  TextBox5 = Tables.Count
End Sub
```

Пример более сложной программы в Word, рассчитывающей значения функции $Y = F(X)$ для X, изменяющегося от $X_{нач.}$ до $X_{кои.}$ и для заданного числа точек, показывающей таблицу и диаграмму, приведен в учебнике [1, п. 5.4.2.1].

Учебное задание к лабораторной работе № 1

1) Создать новый документ Word, в нем запустить запись макроса и написать в документе, чем отличаются правила образования идентификаторов и описания переменных в языке VBA от языка VBS. Сохранить макрос для данного документа. Выполнить макрос 3 раза. Просмотреть текст макроса в окне Microsoft Visual Basic For Applications, скопировать его в свой документ.
2) Создать в том же документе Word четыре поля для работы с данными (элементы ActiveX) следующего вида:

Для второго и третьего полей создайте для события Change обращение к процедуре Расчет_стоимости, которая будет присваивать четвертому полю значение произведения второго поля на третье, как показано выше. Для четвертого поля задайте значение свойства Locked = True, чтобы оно стало недоступным для редактирования.

Лабораторная работа №2. Программирование на языке VBA в Microsoft Office Excel

Главный объект системы – Excel.Application, он имеет множество свойств и методов, отличных от Word.Application, некоторые из них показаны на рисунке 2.1.

Рисунок 2.1 – Некоторые свойства и методы объекта Excel.Application

Объект **Application** имеет важнейшие свойства:

- **Cells** – ячейка активного листа,
- **Range** – одна ячейка или группа ячеек активного листа (вместо свойства Range можно использовать написание адреса или диапазона ячеек в квадратных скобках),
- **Columns** – вертикальная колонка ячеек активного листа,
- **Rows** – горизонтальный ряд ячеек активного листа,
- **Sheets** – свойство, которое возвращает ссылку на коллекцию, состоящую из объектов **Sheet** – множество листов книги со всеми их внедренными объектами (например, диаграммами);
- **Worksheets** – свойство, которое возвращает ссылку на коллекцию, состоящую из объектов **Worksheet** – листов книги без внедренных объектов,
- **Selection** – ссылка на активный выделенный объект (лист, диапазон ячеек, диаграмму (***Chart***), ряд, столбец и пр.

При написании программы в редакторе Microsoft Visual Basic системы Excel в проекте открытого файла (который в этом случае должен сохраняться, как «Книга Excel с поддержкой макросов» с расширением *.xlsm) при использовании методов и свойств можно опускать название объекта **Application**. Методы программирования с использованием этих объектов рассмотрены далее на примерах.

Так же, как и в системе Microsoft Word, основы программирования в VBA для системы Excel можно освоить, воспользовавшись методом записи последовательности своих действий – **Запись макроса** на вкладке **Разработчик**, причем записываются не только нажатия клавиш, но и операции, выполняемые мышкой. Текст макроса позволяет лучше понять те операции, которые начинающий пользователь выполняет иногда не вполне осознанно, например, щелчок мышкой на ячейке А2 – команда `Range("A2").Select` и т. п. Конечно же, предполагается, что пользователь знает некоторые английские слова (например, Range – диапазон , Select – выбрать, Selection – выбор, Font – шрифт, Size – размер, Sheet – лист, Cell – ячейка, Characters – символы, Active – активный, Formula – формула , Center – центр, Border – граница, Edge – грань, Alignment – выравнивание …) и основы работы с объектами, тогда текст программы достаточно легко читается. Макрос будет содержать те приемы работы с объектами Excel, которые рекомендует к использованию разработчик данной системы, хотя возможны и другие варианты.

В учебнике [1] показано, как выполнить расчет значений функции и построение диаграммы Y = F(X), аналогичный приведенному ранее для Word с использованием записи макросов и последующим редактированием VBA-программы.

Запишем Макрос более простой задача – автозаполнения столбца в таблице арифметической прогрессией с шагом 1. Его текст:

```
Sub Макрос1()
   ActiveCell.FormulaR1C1 = "1"  'пишем 1 в A1
   Range("A1:A10").Select        'выделяем диапазон A1:A10
   Selection.DataSeries Rowcol:=xlColumns, _
   Type:=xlLinear,Date:=xlDay, Step:=1, Trend:=False
      'заполняем диапазон прогрессией
End Sub
```

Вместо свойства объекта **Range** (объект – обычно активный рабочий лист **ActiveSheet**) для работы с ячейками можно использовать свойство **Cells**(№_строки, №_столбца). Свойство **Offset** позволяет выбрать ячейку, расположенную на заданное количество позиций по вертикали и горизонтали от текущей. Данное свойство использует синтаксис. <объект>.**Offset**(смещение_строки, смещение_столбца), применяется только к объекту типа **Range.**

Модифицируем Макрос1 таким образом, чтобы он заносил арифметическую прогрессию в любой столбец на листе начиная с выбранной ячейки:

```
Sub Макрос1()
  ActiveCell.Formula = "1"
  adr1 = ActiveCell.Address
  adr2 = ActiveCell.Offset(9, 0).Address
        'адрес со смещением вниз на 9 строк
  Range(adr1, adr2).Select
  Selection.DataSeries Rowcol:=xlColumns, Type:=xlLinear, _
     Date:=xlDay, Step:=1, Trend:=False
End Sub
```

Достаточно важная задача при работе в системе Excel – проверка вводимых в ячейки данных на соответствие некоторым условиям и занесение данных в ячейки из заранее подготовленных списков. Эти задачи можно решить, как с использованием пункта Проверка данных вкладки ленты Данные, так и с использованием программирования на VBA.

Пример: при вычислении по формуле

$$z = \frac{\sqrt{x}}{y}$$

X не может быть отрицательным и Y не может быть равен нулю.

Если не предусмотреть проверку, при вводе нулевого значения Y в ячейке для Z появится сообщение о делении на ноль (рисунок 2.2).

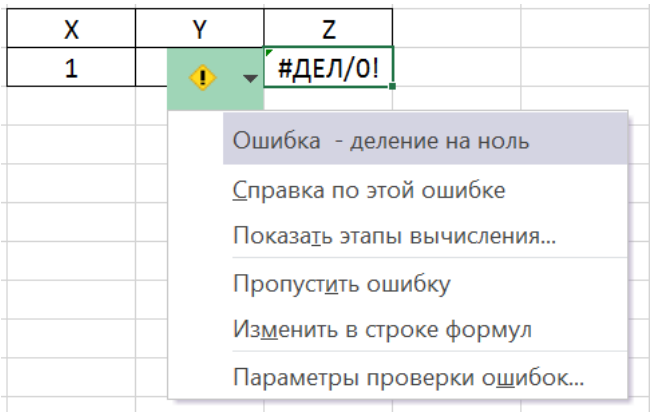


Рисунок 2.2 – Системное сообщение об ошибке

Если воспользоваться Пунктом ленты Проверка данных с заданием в параметрах условия Значение не равно 0 и заданием Вид - Останов, сообщение об ошибке появится в окне сообщений, как показано на рисунке 2.3.

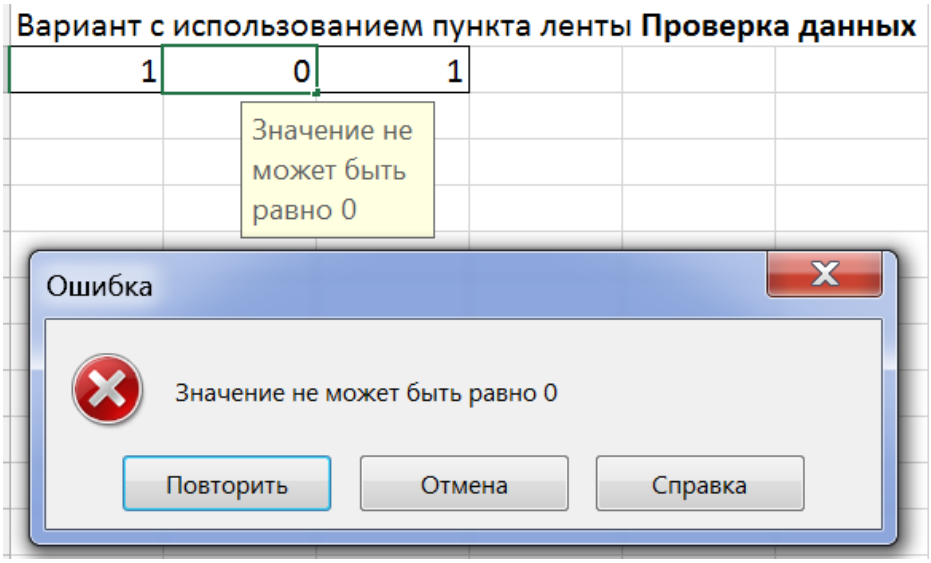
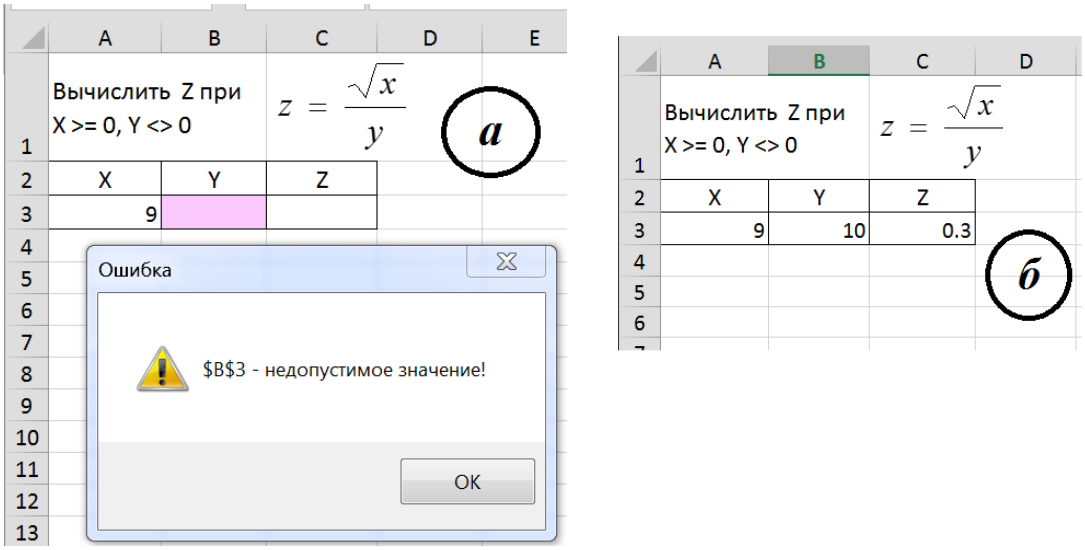


Рисунок 2.3 – Сообщение об ошибке с использованием Проверки данных

Аналогично можно задать проверку для значения X.

Решим эту же задачу с использованием программирования. При этом ячейку с неверным вводом данных будем очищать и выделять розовой заливкой, значение в ячейке для Z будем удалять при неверном вводе X или Y, позицию курсора на листе будем оставлять в ячейке для X или Y, пока не будет задано правильное значение. Основную программу напишем для события листа Change, фиксацию курсора в B3 – для события листа SelectionChange. На рисунке 2.4 показаны результаты реализации этих задач.



а – сообщение при вводе нуля в B3;
б – после задания допустимого значения в B3

Рисунок 2.4 – Работа программы

Текст программ с комментариями:

```
Private Sub Worksheet_Change(ByVal Target As Range)
        'было выполнено редактирование для ячейки Target
 Application.EnableEvents = False
       'отключаем обработку событий,
        'т. к. внутри процедуры есть редактирование ячеек
 Adr = Target.Address       'адрес отредактированной ячейки
If (Adr = "$A$3" Or Adr = "$B$3") Then
       ' если отредактированы A3 или B3
  If [A3].Value < 0 Or [B3].Value = 0 And _
    Range(ADR).Interior.Color <> RGB(250, 200, 250)
       'если значение в ячейке A3 < 0 или B3 = 0 и не розовая ячейка ADR
     [C3].ClearContents         ' очищаем ячейку C3
     Range(ADR).ClearContents     ' очищаем ячейку ADR
     Range(ADR).Interior.Color = RGB(250, 200, 250)
           ' задаем розовый цвет для ADR,
          'чтобы повторно не появлялось сообщение
      MsgBox ADR & " - недопустимое значение!", _
          vbExclamation, "Ошибка"
   Else
     If IsEmpty(Range("c3").Value) Then
       Range("C3").Formula = "=SQRT($A$3)/$B$3"
      Range(ADR).Interior.Pattern = xlNone  'удаляем заливку ADR
     End If
```

```
    End If
  End If
  Application.EnableEvents = True
End Sub

Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  If IsEmpty([A3].Value) Then [A3].Select
    ' возвращаемся в A3, если пустое значение в этой ячейке
  If IsEmpty([B3].Value) Then [B3].Select
    ' возвращаемся в B3, если пустое значение в этой ячейке
End Sub
```

Пункт **Проверка данных** вкладки ленты **Данные** можно использовать также для работы со списками. Список может быть оформлен на том же листе или на другом листе книги *Excel*. Пример использования показан на рисунке 2.5.

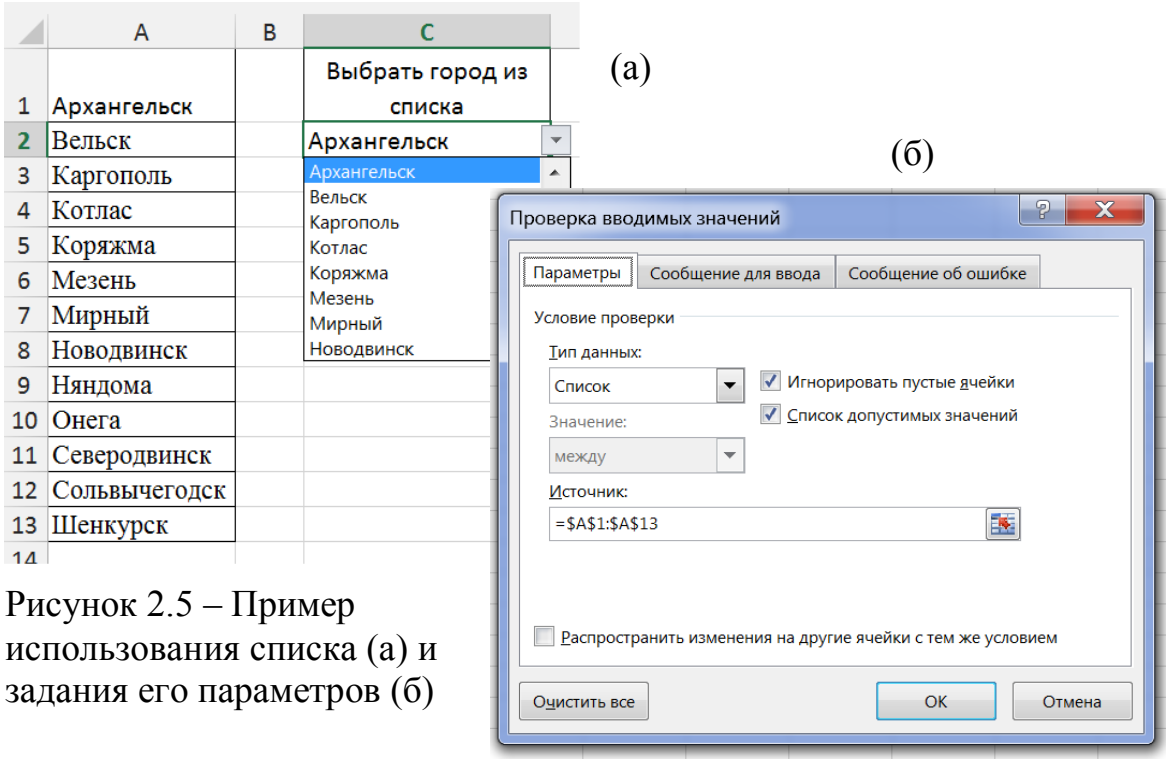


Рисунок 2.5 – Пример использования списка (а) и задания его параметров (б)

Другой пример программирования на языке VBA – расчет заданного количества случайных чисел X в заданном диапазоне показан на рисунке 2.6. Задача дополнена их графическим отображением, поиском минимального X_{min} и максимального X_{max} значений среди этих чисел, и расчетом $Y = X/X_{max}$.

	A	B	C	D	E	F	G	H	I
1	Генерация случайных чисел и расчет Y = X/X_{max}								
2	Исходные данные			Расчет					
3	$X_{нач.}$	$X_{кон.}$	Кол-во	X_{min}	X_{max}				
4	-5.0000	5.0000	15	-4.7212	4.9367				
5	X	Y							
6	-0.8822	-0.1787							
7	-3.0858	-0.6251							
8	2.0291	0.4110							
9	-0.7026	-0.1423							
10	-4.7212	-0.9563							
11	-2.8993	-0.5873							
12	2.9615	0.5999							
13	-0.4749	-0.0962							
14	3.0211	0.6120							
15	-3.9019	-0.7904							
16	-3.2031	-0.6488							
17	3.0361	0.6150							
18	4.9367	1.0000							
19	0.5050	0.1023							
20	0.6146	0.1245							

Рисунок 2.6 – Расчет случайных чисел в заданном диапазоне

Текст главной программы написан для события листа Change, расчет случайных чисел – процедура с именем Расчет, построение диаграммы – процедура с именем График:

```
Private Sub Worksheet_Change(ByVal Target As Range)
  Adr = Target.Address
  If Adr = "$A$4" Or Adr = "$B$4" Or Adr = "$C$4" Then
       ActiveSheet.Unprotect
       Call Расчет
       Call График
       [A4].Select
       ActiveSheet.Protect DrawingObjects:=True, _
            Contents:=True
   End If
End Sub
'------- далее общая процедура в секции General  -----
Sub Расчет()
  n = [C4]  ' количество чисел
  adr2 = "A6:A" & (5 + n)
    ' определяем диапазон для чисел, количество которых=C4
  Randomize
  Range("A6:A105").ClearContents ' очищаем старые значения
  Range("A6:A105").NumberFormat = "0.0000" ' задаем формат
```

```
  For I = 1 To n ' цикл вычисления случайных чисел
    Adr = "A" & I + 5
    Range(Adr).Value = [B4] - ([B4] - [A4]) * Rnd
  Next
  xmin = [A6]
  xmax = [A6]
  For I = 1 To n  ' цикл поиска минимума и максимума
     Adr = "A" & I + 5
     xt = Range(Adr).Value
     If xt < xmin Then xmin = xt
     If xt > xmax Then xmax = xt
  Next
  [D4] = xmin
  [E4] = xmax
For I = 1 To n   ' цикл расчета Y
    Adr1 = "A" & I + 5
    Adr2 = "B" & I + 5
    Range(Adr2).Value = (Range(Adr1).Value) / xmax
  Next
End Sub

Sub График()
   ActiveSheet.Unprotect
   If ActiveSheet.Shapes.Count > 0 Then
     ' удаляем старый график
     ActiveSheet.ChartObjects("Grafik").Activate
     ActiveChart.Parent.Delete
   End If
    Rng = "A6:A" & (5 + [C4])   ' диапазон исходных данных
    ActiveSheet.Shapes.AddChart2(332, xlLineMarkers).Select
    ActiveChart.SetSourceData Source:=Range(Rng)
    ActiveChart.ChartTitle.Select
    ActiveChart.ChartTitle.Text = "Случайные числа"
    ActiveChart.Parent.Name = "Grafik"
    ActiveChart.Axes(xlValue).Select
    Selection.TickLabels.NumberFormat = "0"
    ActiveSheet.Shapes("Grafik").Left = 120
    ActiveSheet.Shapes("Grafik").Top = 80
    ActiveSheet.Shapes("Grafik").Width = 300
    ActiveChart.Axes(xlValue).HasTitle = True
    ActiveChart.Axes(xlValue, xlPrimary). _
       AxisTitle.Text = "Y"
    ActiveChart.SetElement _
```

```
    (msoElementPrimaryValueAxisTitleHorizontal)
  ActiveChart.Axes(xlCategory).HasTitle = True
  ActiveChart.Axes(xlCategory, xlPrimary). _
    AxisTitle.Text = "X"
  ActiveChart.Legend.Select
  Selection.Delete
End Sub
```

Текст программ должен быть достаточно понятен для студентов, ранее изучавших язык Basic и понимающих принципы работы с объектами.

Учебное задание к лабораторной работе № 2

1) Создать новый файл Excel с поддержкой макросов. Запустить запись макроса для данного документа, в ячейку A1 занести число 1,11; в ячейку B1 занести число 2,22; в ячейке C1 записать формулу =A1*B1. Остановить запись макроса. Удалить данные из ячеек A1:C1 и выполнить сохраненный макрос. Изучить его текст в окне Microsoft Visual Basic For Applications, скопировать его текст на лист Excel. Модифицировать макрос для работы с любой начальной ячейкой.
2) Двумя способами (с использованием *Проверки данных* и с написанием VBA-программы) выполнить проверку правильности ввода (выполнить вариант в таблице, соответствующий номеру компьютера):

Вариант №	Расчетная формула	Условия проверки
1	$y = \frac{\sqrt{a}}{2b}$	a >= 0, b <> 0
2	$y = \sqrt{a+b} + \frac{b}{5-a}$	a + b >= 0; a <> 5
3	$y = \lg(a) + \sqrt{2+b}$	a > 0, b >= –2
4	$y = \frac{\sqrt{2a+3}}{\lg(2b-3)}$	a >= 1,5; b > 1,5
5	$y = \frac{\lg(10-b)}{a}$	a <> 0, b < 10
6	$y = \frac{a^3}{a-b}$	a <> b

7	$y = \frac{2a + b}{b\sqrt{\lg(a)}}$	a > 1 b<>0
8	$y = \sqrt{(1-a)/b}$	a < 1; b > 0
9	$y = \frac{1}{\sqrt{\frac{14-b}{2a}}}$	a > 0 b < 14
10	$y = \frac{2\sqrt{a}}{2-b}$	a >= 0 b <> 2

3) Для заданного количества **n** случайных чисел (как на рисунке 2.6) написать программу на языке VBA для вычисления по заданной формуле (выполнить вариант, соответствующий номеру компьютера):

1)	$y_i = x_i \sum_{\substack{i=1 \\ j=n}}^{\substack{1 \\ n}} x_i \cdot x_j$	2)	$y_i = x_i \sum_{i=1}^{n} x_i$
3)	$y_i = \frac{x_i}{\sum_{i=1}^{n} x_i}$	4)	$z_i = \frac{(n-i) \cdot x_i}{y_i}$
5)	$y_i = \frac{x_i}{x_{\max} - x_{\min}}$ $x_{\max}$, $x_{\min}$ - максимальное и минимальное значения x	6)	$y_i = \frac{x_i}{x_{cp}}$ $x_{cp} = \sum_{i=1}^{n} x_i / n$
7)	$y_i = x_i \sum_{i=1}^{n} x_i / \mathrm{i}$	8)	$y_i = \frac{x_i}{\sum_{i=1}^{n} x_i}$
9)	$y_i = ax_i^2 + bx_i + c$	10)	$y_i = x_i \sum_{i=1}^{n-1} \frac{x_i}{x_{i+1}}$

Лабораторная работа № 3. Программирование на языке VBA в Microsoft Office Access

В Microsoft Office существует объект Access.Application. С использованием его можно запустить в работу Access, его свойства и методы позволяют настроить параметры пользовательского интерфейса, получить доступ к некоторым групповым операциям с таблицами, к компонентам форм и отчетов, к программным модулям и пр.

При разработке клиентских приложений для работы с базой данных в виде экранных форм системы Access могут использоваться методы и свойства этого объекта (формы) и всех вложенных в него объектов (прежде всего полей) с использованием библиотеки *Microsoft Access Object Library.*

Однако, для выполнения сложных расчетных операций с данными множества записей таблицы базы данных в системе Access необходимо использовать библиотеку **ADO** (*Microsoft ActiveX Data Objects)* или **DAO** (*Microsoft Data Access Objects)*, обеспечивающих подключение к базам данных и выполнение операций с данными их таблиц из самых разных программных систем. Библиотека **ADO** более современная, чем **DAO**, считается, что она содержит более развитые возможности работы с данными. Основы работы в этих двух системах очень близки, главным объектом в них на уровне базы данных является ***Recordset*** (набор записей) – временная таблица (**cursor** – курсор), созданная из записей таблицы базы данных или в результате выполнения запроса.

При создании объекта ***Recordset*** могут быть определены динамические или статические его типы (см. таблицу 3.1 для библиотеки **ADO**).

Таблица 3.1 – Типы объекта ***Recordset*** в библиотеках **ADO**

Тип	Краткое описание
Dynamic cursor	Позволяет видеть добавление, изменение, удаление записей, выполняемые другими пользователями. Метод <**Recordset**>.**Update** позволяет обновить записи в таблице базы данных.
Keyset cursor	Позволяет видеть изменение записей, не позволяет видеть добавленные записи, не позволяет редактировать удаленные записи, при выполнении этих операций другими пользователями.

Продолжение таблицы 3.1

Тип	Краткое описание
Static cursor	Статическая копию ряда записей, которую можно использовать, чтобы найти данные или генерировать отчеты. Дополнения, изменения, или удаления записей, выполняемые другими пользователями, не будут видимы. Это – единственно возможный тип курсора, который можно создать на стороне клиента.
Forward-only cursor	Позволяет перемещаться только вперед в объекте **Recordset**. Добавления, изменения или удаления записей другими пользователями не видимы. Используется, когда необходимо однократное прохождение по записям объекта **Recordset**.

Для объекта ***Recordset*** как в **ADO**, так и в **DAO** определено большое количество свойств и коллекций, методов и событий, краткое описание их для **ADO. *Recordset*** приведено в **Приложении.**

Пример работы с объектами Access и использования возможностей библиотек **ADO** и **DAO** рассмотрен в учебнике [1] на примере достаточно сложного расчета сдельного наряда.

Рассмотрим здесь более простые примеры программирования при работе с базами данных в системе Access. Напомним здесь же вкратце процесс создания таблиц базы данных и Windows-форм.

Пример 1. Необходимо разработать фрагмент информационной системы для учета прихода товаров на склад в соответствии с типовой межотраслевой формой № М-4 (утверждена постановлением Госкомстата России от 30.10.97 № 71а), показанной на рисунке 3.1.

Для решения поставленной задачи необходимо создать как минимум 3 таблицы: Список ордеров, Товары ордера (со связью между ними по номеру ордера) и Справочник структурных подразделений. Для упрощения примера не будем показывать полную структуру первой таблицы (таблица 3.1) и третьей таблиц (таблица 3.3) и покажем полную структуру второй таблицы (таблица 3.2).

Для таблицы Список ордеров предусмотрим сохранение итоговых сумм, т. к. эти данные в организации могут понадобиться для формирования сводных отчетов.

После описания структуры таблиц с использованием Конструктора таблиц устанавливаем связи между таблицами (рисунок 3.2) и разрабатываем форму для работы с информацией базы данных (рисунок 3.3).

ПРИХОДНЫЙ ОРДЕР № 111

	Коды
Форма по ОКУД	0315003
по ОКПО	

Организация: Лесозавод 1

Структурное подразделение: Склад 1

Дата составления	Код вида операции	Склад	Поставщик: наименование	Поставщик: код	Страховая компания	Корреспондирующий счет: счет, субсчет	Корреспондирующий счет: код аналитического учета	Номер документа: сопроводительного	Номер документа: платежного	
20.03.2013	1	1	Цех 1	1	-			накл. №28	456	

Материальные ценности: наименование, сорт, размер, марка	Материальные ценности: номенклатурный номер	Единица измерения: код	Единица измерения: наименование	Количество: по документу	Количество: принято	Цена, руб. коп.	Сумма без учета НДС, руб. коп.	Сумма НДС, руб. коп.	Всего с учетом НДС, руб. коп.	Номер паспорта	Порядковый номер по складской картотеке
1	2	3	4	5	6	7	8	9	10	11	12
Доска обрезная 25*100*6000	25		$м^3$	1.500	1.500	4 800.00	7 200.00	1 296.00	8 496.00		
Доска обрезная 40*100*6000	25		$м^3$	1.000	1.000	5 200.00	5 200.00	936.00	6 136.00		
Доска обрезная 50*100*6000	25		$м^3$	2.000	2.000	5 500.00	11 000.00	1 980.00	12 980.00		
Оборотная сторона формы											
				Итого	4.500	×	23 400.00	4 212.00	27 612.00		

Принял кладовщик (должность) _____ (подпись) Иванов П. С. (расшифровка подписи) **Сдал** экспедитор (должность) _____ (подпись) Петров И. И. (расшифровка подписи)

Рисунок 3.1 – Форма № М-4 учета поступления товаров на склад

Таблица 3.2 – Структура таблицы Список ордеров

Имя поля	Тип данных	Размер поля	Индексированное поле
№ ордера	Числовой	Длинное целое	Да, **Ключевое поле**
Структурное подразделение	Числовой	Целое	Да (Допускаются совпадения)
Дата	Дата/время	Краткий формат даты	Нет
Код вида операции	Числовой	Байт	Нет
Склад	Текст	127	Нет
Поставщик	Текст	127	Нет
Поставщик код	Текст	15	Нет
Принял должность	Текст	25	Нет
Принял расшифровка подписи	Текст	25	Нет
Сдал должность	Текст	25	Нет
Сдал расшифровка подписи	Текст	25	Нет
Количество всего	Числовой	Одинарное с пл. точкой	Нет

Продолжение таблицы 3.2

Имя поля	Тип данных	Размер поля	Индексированное поле
Сумма без НДС всего	Числовой	Одинарное с пл. точкой	Нет
Сумма НДС всего	Числовой	Одинарное с пл. точкой	Нет
Сумма с НДС всего	Числовой	Одинарное с пл. точкой	Нет

Примечание. Для всех полей с типом Числовой - Одинарное с пл. точкой задан формат поля С разделителем разрядов и Число десятичных знаков = 2.

Таблица 3.3 – Структура таблицы Товары ордера

Имя поля	Тип данных	Размер поля	Индексированное поле
№ ордера	Числовой	Длинное целое	Да (Допускаются совпадения)
Наименование, сорт, размер, марка	Текст	127	Нет
Номенклатурный номер	Текст	25	Нет
Код ед-цы измерения	Текст	25	Нет
Наименование ед-цы измерения	Текст	127	Нет
Количество по документу	Числовой	Одинарное с пл. точкой	Нет
Количество принято	Числовой	Одинарное с пл. точкой	Нет
Цена, руб коп	Числовой	Одинарное с пл. точкой	Нет
Сумма без учета НДС, руб коп	Числовой	Одинарное с пл. точкой	Нет
Сумма НДС, руб коп	Числовой	Одинарное с пл. точкой	Нет
Всего с учетом НДС руб коп	Числовой	Одинарное с пл. точкой	Нет
Номер паспорта	Текст	25	Нет
Порядковый номер по складской картотеке	Текст	15	Нет

Таблица 3.4 – Структура таблицы Справочник структурных подразделений

Имя поля	Тип данных	Размер поля	Индексированное поле
Структурное подразделение	Числовой	Целое	Да, **Ключевое поле**
Наименование	Текст	127	Нет

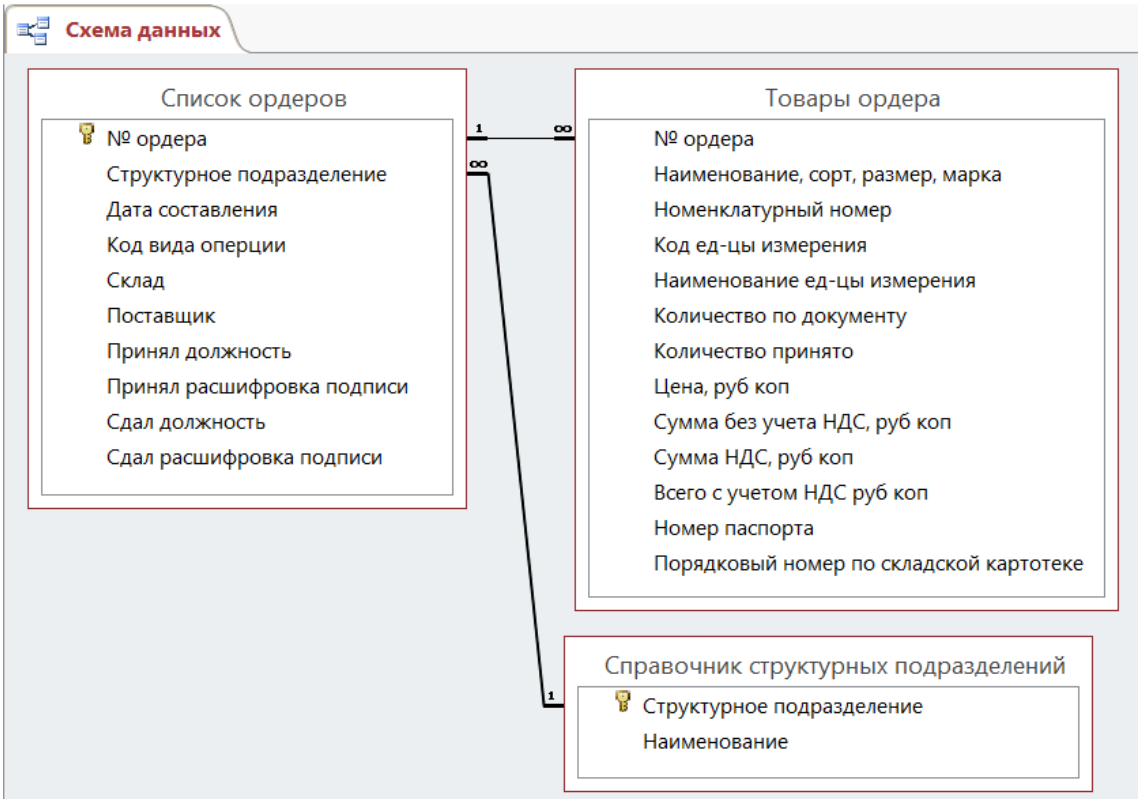


Рисунок 3.2 – Схема данных

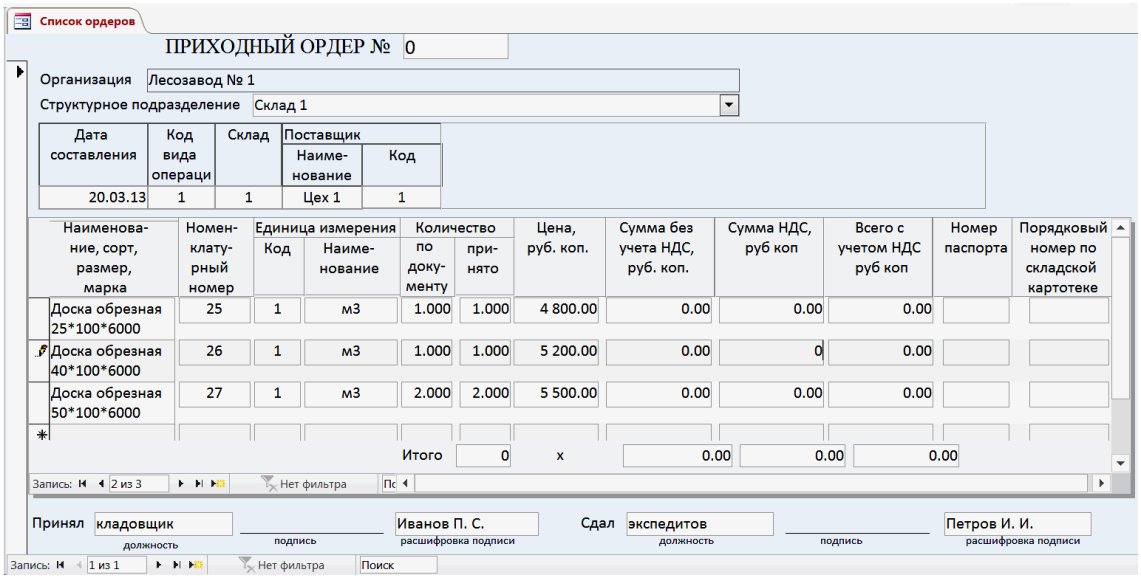


Рисунок 3.3 – Форма для работы с информацией базы данных

На рисунке 3.3 форма содержит как поля для работы с исходными данными, так и расчетные поля Сумма…, Всего и Итого, для вычисления которых используются следующие программы на языке VBA:

```
Option Compare Database
Sub Расчет()
 Me.Сумма_без_учета_НДС__руб_коп = _
      Me.Количество_принято * Me.Цена__руб_коп
 snds = Me.Сумма_НДС__руб_коп
 otv = MsgBox("Пересчитать НДС по ставке 18%?", _
          vbYesNo, "Вопрос к пользователю:")
 If otv = 6 Then  ' ответ Да
   Me.Сумма_НДС__руб_коп = _
      Me.Сумма_без_учета_НДС__руб_коп * 0.18
 End If
 Me.Всего_с_учетом_НДС_руб_коп = _
  Me.Сумма_без_учета_НДС__руб_коп +
Me.Сумма_НДС__руб_коп
 NN = Me.[№ ордера]
 Me.Refresh ' обновить данные формы
 Dim rs As New ADODB.Recordset
 ' текст SQL-запроса:
 sq = "SELECT DISTINCTROW [Товары ордера].[№ ордера], " + _
  "Sum([Товары ордера].[Сумма без учета НДС, руб коп]) " + _
  "AS sbn, Sum([Товары ордера].[Сумма НДС, руб коп])  " + _
  "AS sn, Sum([Товары ордера].[Всего с учетом НДС руб коп]) " + _
  "AS ssn, Sum([Товары ордера].[Количество принято]) " + _
  "AS sk FROM [Список ордеров] INNER JOIN [Товары ордера] "
+ _
  "ON [Список ордеров].[№ ордера] = [Товары ордера].[№ ордера] "
+ _
  "GROUP BY [Товары ордера].[№ ордера], " + _
  "[Список ордеров].[№ ордера]  " + _
  "HAVING ((([Товары ордера].[№ ордера])= " & NN & "))"
  rs.Open sq, CurrentProject.Connection, adOpenDynamic, _
    adLockOptimistic
' присваиваем полученные в запросе значения
' полям формы главной таблицы Список ордеров
  Me.Parent.Количество_всего = rs!sk
  Me.Parent.Сумма_без_НДС_всего = rs!sbn
  Me.Parent.Сумма_НДС_всего = rs!sn
  Me.Parent.Сумма_с_НДС_всего = rs!ssn
  Me.Refresh
```

```
End Sub

Private Sub Количество_принято_AfterUpdate()
  If Me.Цена__руб_коп > 0 Then
    Call Расчет
     ' рассчитываем суммы в строке таблицы товаров
    Me.Количество_принято.SetFocus
  End If
End Sub


Private Sub Сумма_без_учета_НДС__руб_коп_AfterUpdate()
  Call Расчет
End Sub

Private Sub Сумма_НДС__руб_коп_AfterUpdate()
  If IsNull(Me.Сумма_НДС__руб_коп) Then _
     Me.Сумма_НДС__руб_коп = 0
  Call Расчет
End Sub

Private Sub Цена__руб_коп_AfterUpdate()
  If Me.Количество_принято > 0 Then
    Call Расчет
    Me.Цена__руб_коп.SetFocus
  End If
End Sub
```

В этом тексте все названия полей после ***Me.*** выбираются из всплывающего списка, как показано на рисунке 3.4 (более корректно было бы после каждого имени приписавать свойство ***.Value***, но система прекрасно работает со значениями полей с указанием только их имени, как показано в программе).

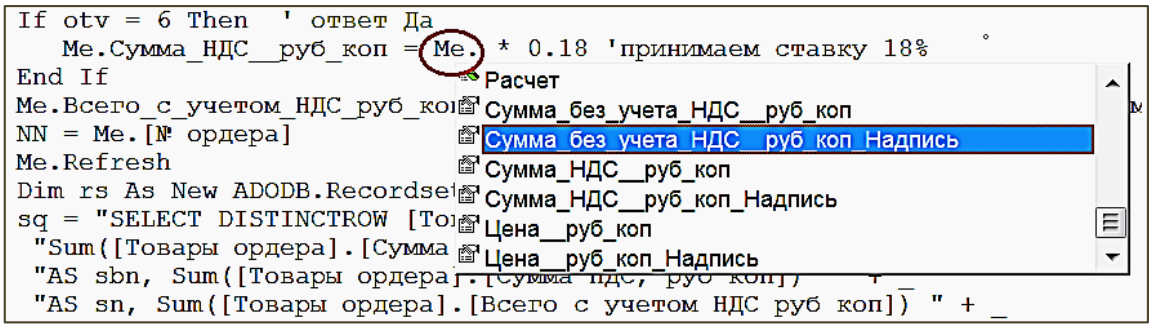


Рисунок 3.4 – Всплывающие списки в редакторе VBA

Текст SQL-запроса можно получить с использованием режима *Создание – Запросы – Мастер запросов* в системе Access.

Расчет будет выполняться для событий **`AfterUpdate`** (после обновления) для полей формы [Количество принято], [Сумма без учета НДС, руб коп], [Сумма НДС, руб коп] и [Цена, руб коп].

Вид формы после расчета показан на рисунке 3.5.

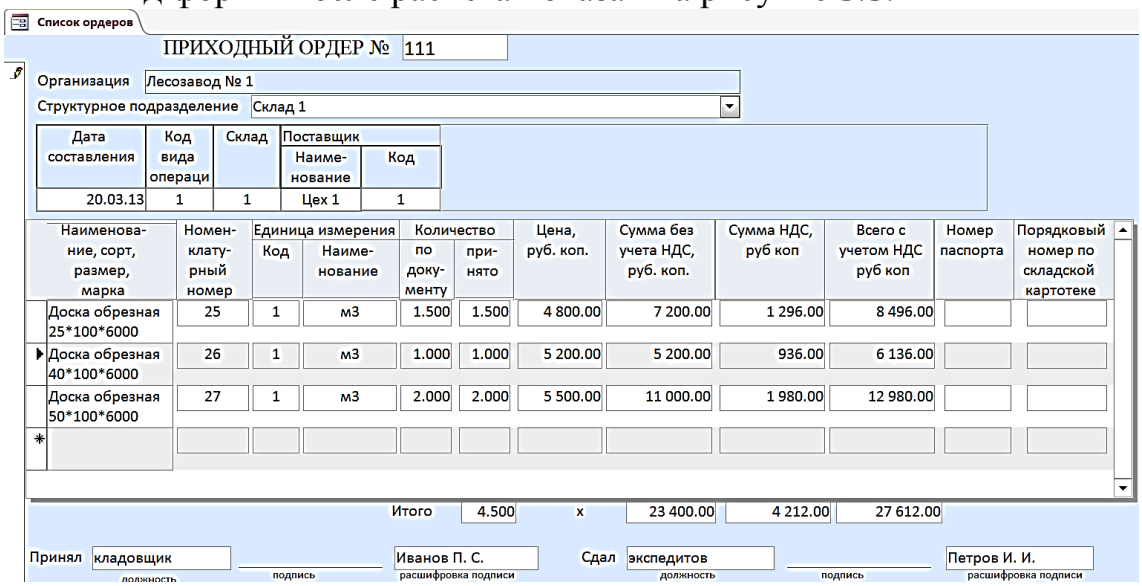


Рисунок 3.5 – Форма после расчета по программе

Еще один пример – импорт данных в таблицы базы из внешних источников. В системе Access на вкладке ленты Внешние данные – Импорт и связи есть возможность добавить в базу данных информацию из Excel, других баз данных, текстового и XML-файлов, но нет импорта из документов Word. Рассмотрим такую возможность – разработку программы чтения данных из таблиц файлов *.doc или *.docx с перенесением этих данных в существующие таблицы базы данных. Предположим, для предыдущего примера Приходного ордера в документе Word есть 2 таблицы (таблица 3.5).

Таблица 3.5 –Таблицы файла *d:\asg\Приходный ордер.docx*

№ ордера	Дата составления	Структурное подразделение
115	2.03.2013	1

Наименование	Ед-цы из-мерения	Кол-во по док	Цена
Доска необрезная 25х6000	м3	1	3500
Доска необрезная 50х6000	м3	1.5	4800
Вагонка 1145Х20 2.0-6.0	м2	50	300
Евровагонка 12,5х96 2.0-4.0	м2	100	200

Программу импорта данных следует написать, как процедуру обработки события *Нажатие кнопки (Click)* для новой кнопки с именем ***Импорт***, которую можно разместить на существующей форме.

Текст процедуры может быть следующий:

```
Private Sub Импорт_Click()
'On Error Resume Next ' использовать в окончательном варианте
Dim rs1 As New ADODB.Recordset 'переменная типа Recorgset
Dim rs2 As New ADODB.Recordset
SQL1 = "SELECT [Список ордеров].* FROM [Список ордеров]"
SQL2 = "SELECT [Товары ордера].* FROM [Товары ордера]"
        'тексты SQL-запросов – источника данных Recordset
rs1.Open SQL1, CurrentProject.Connection, _
      adOpenDynamic, adLockOptimistic
rs2.Open SQL2, CurrentProject.Connection, _
      adOpenDynamic, adLockOptimistic
       ' создаем динамические курсоры для таблиц Список ордеров
       ' и Товары ордера с оптимистической блокировкой записей
Set objWord = CreateObject("Word.Application")
        'создаем объект типа Word.Application
'objWord.Visible = True  ' – только до завершения отладки
Set objDoc = _
  objWord.Documents.Open("d:\asg\Приходный ордер.docx ")
          'открываем файл для объекта objWord
N_ord = objDoc.Tables(1).Cell(2, 1).Range.Calculate
Data_ord = objDoc.Tables(1).Cell(2, 2)
Data_ord = CDate(Left(Data_ord, Len(Data_ord) - 2))
Str_podrazd = objDoc.Tables(1).Cell(2, 3).Range.Calculate
With rs1
  .AddNew      'добавляем строку в курсор
  .Fields("№ ордера") = N_ord
  .Fields("Дата составления") = Data_ord
  .Fields("Структурное подразделение") = Str_podrazd
  .Update     ' пытаемся сохранить курсор в таблице базы
   If Err.Number <> 0 Then 'если сохранить не удается
     otv = MsgBox("Есть в базе данных базу Ордер с № " _
     & N_ord & ". Добавить товар в этот приходный ордер?", _
     vbYesNo + vbExclamation, "Есть такой приходный ордер!")
      If otv = 7 Then   ' нажата кнопка НЕТ
        rs1.Close
        rs2.Close
        objDoc.Close
        objWord.Quit
```

```
        Exit Sub              ' выход из процедуры
      End If
    End If
End With
n = objDoc.Tables(2).Rows.Count 'определяем количество строк
                                'в таблице 2 документа Word
no_dat = ""   ' строка для записи неудачных операций.Update
yes_dat = ""  ' строка для записи удачных операций.Update
For i = 2 To n
   s = objDoc.Tables(2).Rows(i)
            ' i-я строка таблицы 1 документа
   fld = Split(s, Chr(13) + Chr(7))
      ' i-тую строку таблицы разбиваем на поля для массива fld
   With rs2  ' далее к методам и свойствам объекта rs
             ' можно обращаться, начиная с точки
   .AddNew 'добавить запись в курсор
   .Fields("№ ордера") = N_ord
   .Fields("Наименование, сорт, размер, марка") = fld(0)
   .Fields("Наименование ед-цы измерения") = fld(1)
   .Fields("Количество по документу") = Eval(fld(2))
   .Fields("Цена, руб коп") = Eval(fld(3))
   .Update     ' команда сохранить курсор в таблице базы
     If Err.Number <> 0 Then   'если сохранить не удается
       no_dat = no_dat & fld(0) & vbLf
     Else
      yes_dat = yes_dat & fld(0) & vbLf
     End If
   End With
  Next
rs.Close
objDoc.Close
objWord.Quit
MsgBox "Всего в табл. 2 Word " & n & " строк " & vbLf & _
 "В Приходный ордер № " & N_ord & vbLf & _
 "в таблицу [Товары ордера] добавлены записи:" & vbLf & _
  yes_dat & _
 "Не добавлены из-за нарушений целостности базы::" _
  & vbLf & no_dat, , "Результаты переноса данных"
rs1.Close
rs2.Close
objDoc.Close
objWord.Quit
```

```
Me.Requery
Me.Recordset.MoveLast
End Sub
```

Результаты выполнения программы будут показаны в окне MsgBox (см. рисунок 3.6).

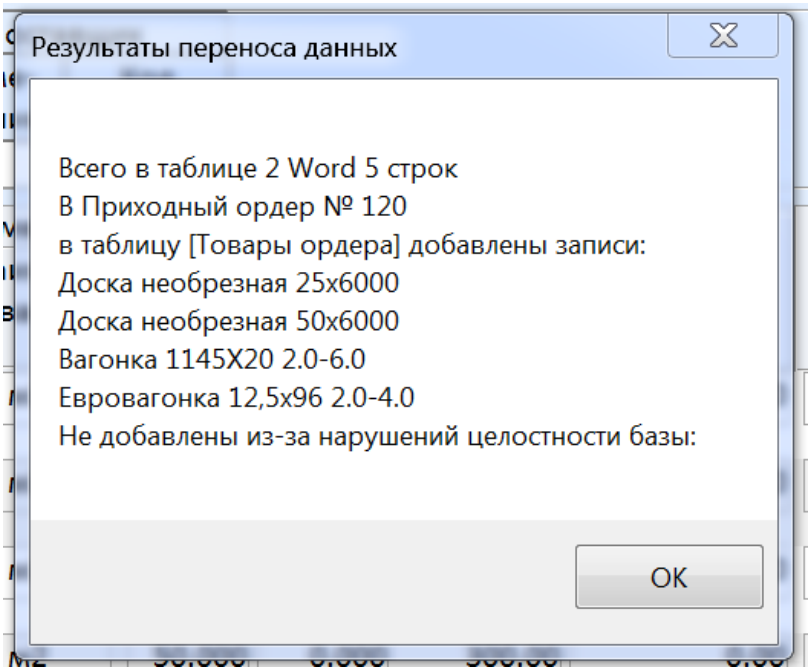


Рисунок 3.6. Сообщение после выполнения процедуры добавления записей в таблицу Access из таблицы Word

Учебное задание к лабораторной работе № 3

Задание выдается преподавателем индивидуально в виде распечатки форм для проектирования информационной системы или в виде файла с изображениями форм.

Приложение. Методы и свойства объекта ADO.Recordset

Свойства/Коллекции

Имя	Краткое описание
AbsolutePage	Указывает, на какой странице текущая запись находится
AbsolutePosition	Указывает порядковую позицию текущей записи в объекте Recordset
ActiveCommand	Указывает объект *Command*, созданный связанным объектом Recordset
ActiveConnection	Указывает, какому объекту *Connection* в настоящее время принадлежат указанная команда, Recordset, или запись
BOF, EOF	**BOF** — указывает, что текущая позиция - перед первой записью в объекте Recordset. **EOF** — указывает, что текущая позиция - после последней записи
Bookmark	Закладка, которая однозначно определяет текущую запись в объекте Recordset или устанавливает текущую запись в объекте Recordset на запись, идентифицированную этой значением закладки *Bookmark*
CacheSize	Указывает число записей объекта Recordset, которые кэшируются в памяти
CursorLocation	Указывает местоположение курсора
CursorType	Указывает тип курсора, используемого в объекте Recordset
DataMember	Указывает Имя компонента данных, который будет найден для объекта, на который ссылается свойство DataSource
DataSource	Указывает объект, который содержит данные, которые будут представлены как объект Recordset
EditMode	Указывает статус редактирования текущей записи
Fields *(Collection)*	Коллекция полей для записи *Recordset*
Filter	Указывает фильтр данных для *Recordset*.
Index	Указывает имя индекса, заданного для объекта *Recordset*
LockType	Указывает тип блокировки
MarshalOptions	Указывает, какие записи должны быть возвращены назад на сервер
MaxRecords	Указывает максимальное количество записей, которое возвращает Recordset для запроса
PageCount	Указывает сколько страниц данных содержит объект *Recordset*
PageSize	Указывает из сколько записей состоит одна логическая страница данных в *Recordset*.
Свойства *(Collection)*	Коллекция свойств объекта
RecordCount	Указывает количество записей в объекте *Recordset*
Sort	Указывает одно имя поля или несколько по которым объект *Recordset* сортируется в порядке возрастания или убывания
Source	Указывает источник данных для объекта Recordset
State	Состояние объекта: открыт, закрыт; соединяется, исполняется, выполняется
Status	Статус текущей записи
StayInSync	Указывает, в иерархическом объекте Recordset, выполняются или нет изменения соответствующих дочерних записей, если позиция родительской строки изменяется

Методы

Имя	Краткое описание
AddNew	Создание новой записи в обновляемом объекте Recordset.
Cancel	Отменить выполнение
CancelBatch	Отменить выполнение пакетного обновления
CancelUpdate	Отменить обновление для объекта *Recordset* или коллекции полей или объекта *Record* до вызова метода *Update*
Clone	Создать дубликат объекта *Recordset*, доступный только для чтения
Close	Закрыть объект и все подчиненные объекты
CompareBookmarks	Сравнивает две закладки и возвращает их относительное положение (CompareEnum)
Delete	Удаляет текущую запись или группу записей
Find	Поиск записи в объекте *Recordset* для заданных условий
GetRows	Переносит множество записей объекта *Recordset* в массив
GetString	Возвращает *Recordset* как строку с разделителями
Move	Перемещает текущую позицию в объекте *Recordset*
MoveFirst, MoveLast, MoveNext, and MovePrevious	Перемещает текущую позицию в объекте *Recordset* в начало, в конец, на следующую или предыдущую запись
NextRecordset	Очищает текущий объект Recordset и возвращает следующий Recordset при продвижении через ряд команд
Open	Открыть курсор
Requery	Обновляет данные в объекте Recordset перезапуском запроса, на котором базируется этот объект
Resync	Обновляет данные в текущем объекте Recordset, или коллекции полей объекта из основной базы данных
Save	Сохраняет Recordset в файле, или объекте *Stream*
Seek	Ищет индекс Recordset, чтобы быстро определить запись, которая соответствует указанным значениям, и изменяет текущую позицию строки в объекте Recordsetё
Supports	Определяет, поддерживает ли указанный объект Recordset специфический тип функциональности
Update	Сохраняет любые изменения, которые Вы делаете в текущей строке объекта Recordset, или коллекции полей объекта *Record*
UpdateBatch	Записывает все пакетные обновления на диск

События (events)

Имя	Краткое описание
EndOfRecordset	Происходит, когда выполняется попытка двигаться за последнюю запись объекта Recordset
FetchComplete	Происходит после того, как все записи в длинной асинхронной операции были найдены в Recordset
FetchProgress	Происходит периодически в течение длинной асинхронной операции, чтобы сообщить, сколько записей было найдено в операции выборки
WillChangeField and FieldChangeComplete	Событие WillChangeField происходит прежде, чем изменяется значение одного или более полей в Recordset. Событие FieldChangeComplete происходит после того, как значение одного или более полей изменилось

WillChangeRecord and RecordChangeComplete	Событие WillChangeRecord происходит перед изменением одной или более записей в Recordset. Событие RecordChangeComplete происходит после одного или более изменений записей
WillChangeRecordset and RecordsetChangeComplete	Событие WillChangeRecordset происходит прежде, чем изменяется Recordset. Событие RecordsetChangeComplete происходит после того, как Recordset изменился
WillMove and MoveComplete	Событие WillMove происходит прежде, чем изменяется текущая позиция в Recordset. Событие MoveComplete происходит после изменения текущей позиции в Recordset
WillMove and MoveComplete	Событие WillMove происходит прежде, чем изменяется текущая позиция в Recordset. Событие MoveComplete происходит после изменения текущей позиции в Recordset

Литература

1) Грошев А. С. Информатика: Учебник для вузов – Архангельск, Арханг. гос. техн. ун-т, 2010. – 470 с.
2) Грошев *А. С. Программирование* на языке Microsoft Visual Basic Scripting Edition: метод.указ.к выполнению лабораторных работ.– Архангельск: АГТУ, 2009. – 82 с.
3) Уокенбах, Джон. Excel 2010: профессиональное программирование на VBA.: Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2012. – 944 с.