Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Санкт-Петербургский горный университет

Кафедра информатики и компьютерных технологий

ИНФОРМАТИКА

ПРОГРАМИРОВАНИЕ НА VBA. ОБРАБОТКА И ВИЗУАЛИЗАЦИЯ ДАННЫХ

Методические указания к лабораторным работам для студентов специальностей 27.03.01, 13.03.02, 20.03.01

САНКТ-ПЕТЕРБУРГ

2019

УДК 004.432.42

ИНФОРМАТИКА. Программирование на VBA. Обработка и визуализация данных : Методические указания к лабораторным работам / санкт-петербургский горный университет. Сост.: *А.Е. Ильин,., С.Ю. Кротова, А.В Чиргин* СПб. 2019. 45 с.

В методическом пособии представлен теоретический материал и практические задания для выполнения лабораторных работ Представлены примеры решения задач по созданию и обработке массивов, созданию и редактированию макросов, рассмотрены варианты построения диаграмм.

Методические указания предназначены для студентов специальностей 27.03.01 «Стандартизация и метрология», 13.03.02, «Электроэнергетика и электротехника», 20.03.01 «Техносферная безопасность».

Табл. 3 Ил. 36 Библиогр.: 5 назв.

Научный редактор доц. *И.И. Пивоварова*

Рецензент: *К. В. Столяров*

Введение

В настоящее время в мире генерируется около 15 Пбайт (1 петабайт ≈ 10^{15} байт) данных ежедневно. Информация удваивается каждые 18 месяцев. По прогнозам к 2025 году объем информации в мире превысит 170 Збайт (1 зеттабайт ≈ 10^{21} байт). Как не утонуть в этом море информации? Как ее хранить, обрабатывать и анализировать?

Данные могут быть полезны при условии, что они будут структурированы так, чтобы аналитик мог с ними работать. В противном случае они могут даже сыграть негативную роль. Известно, что если необходимо «спрятать» полезную информацию от человека, ее можно скрыть в большом потоке неструктурированных данных, и человек будет дезориентирован, «утонет» в этом потоке. Поэтому данные необходимо структурировать. Одними из важнейших структур данных являются вектора и массивы.

Другая проблема связанная с данными - это представление данных в виде, который обеспечивает наиболее эффективную работу человека по их изучению. Этот процесс называется визуализацией данных**.**

Визуализация данных - это отображение больших массивов числовой информации в виде графических объектов. Продукты визуализации данных предназначены для дальнейшей интеграции в информационные системы и системы поддержки принятия решений.

Визуализация данных находит применение в самых разных сферах человеческой деятельности. Для примера назовем медицину (компьютерная томография), научные исследования (визуализация строения вещества, векторных полей и других данных), моделирование тканей и одежды, опытно-конструкторские разработки,

статистика и отчеты и др. Одним из распространенных методов визуализации являются графики и диаграммы.

1. Работа с массивами

Вектор (одномерный массив) - структура данных с фиксированным числом элементов одного и того же типа. Каждый элемент вектора имеет уникальный в рамках заданного вектора номер. Обращение к элементу вектора выполняется по имени вектора и номеру требуемого элемента.

Элементы вектора размещаются в памяти в подряд расположенных ячейках памяти. Под элемент вектора выделяется количество байт памяти, определяемое базовым типом элемента этого вектора. Необходимое число байтов памяти для хранения одного элемента вектора называется слотом. Размер памяти для хранения вектора определяется произведением длины слота на число элементов.

Массив - такая структура данных, которая характеризуется:

- фиксированным набором элементов одного и того же типа;
- каждый элемент имеет уникальный набор значений индексов;
- количество индексов определяют мерность массива, например, два индекса - двумерный массив, три индекса - трехмерный массив, один индекс - одномерный массив или вектор;
- обращение к элементу массива выполняется по имени массива и значениям индексов для данного элемента.

Двумерный массив можно представить как таблицу, а трехмерный – как группу таблиц, каждая из которых имеет одно и то же количество строк и столбцов. Массивы VBA могут иметь до 60 измерений.

Многомерные массивы хранятся в непрерывной области памяти. Размер слота определяется базовым типом

элемента массива. Количество элементов массива и размер слота определяют размер памяти для хранения массива.

Массив с заданным изначально размером называется массивом *фиксированного размера или статическим*. Массив с переменным размером называется *динамическим*.

В начале программы *статический массив* объявляется следующим образом:

Dim *ИмяМассива(*Размерность*)* As *ТипЭлементов*

При этом надо учитывать, что по **умолчанию индексы массива начинаются с 0.**

Например,

Dim A(3) As Integer – одномерный массив A из 4 целых чисел. Элементы этого массива: A(0), A(1), A(2), A(3).

Dim B(3,4) As Single – матрица B из четырех строк и пяти столбцов действительных чисел одинарной точности.

Что бы перейти к общепринятому в математике индексированию элементов массива с 1 надо указать в самом верху модуля директиву

Option Base 1

В этом случае например массив A(3) будет содержать 3 элемента: A(1), A(2) и A(3).

Другой (более предпочтительный) способ указать, что нумерация индексов массива начинается с 1 – использовать описание диапазона индексов с помощью ключевого слова To. Например, чтобы создать массив из пяти элементов типа *Integer*, в котором значение индекса должно изменяться от 1 до 5, используйте такой оператор:

Dim A(1 To 5) As Integer

Динамические массивы в VBA объявляются в начале процедур без указания размерности:

Dim Massiv2() As String

Такие массивы используются, когда заранее неизвестна размерность, которая определяется в процессе выполнения программы. Когда нужная размерность массива становится известна, массив переопределяется с помощью оператора ReDim:

ReDim ArrayName(Subscript),

где Subscript – либо индекс последнего элемента в массиве, либо диапазон индексов (например, 1 To 20).
После переопределения кол-ва элементов массива, все элементы обнуляются. Для того чтобы ***сохранить значения элементов массива,*** после **ReDim** необходимо добавить ключевое слово **Preserve**:

ReDim Preserve MyArray (25).

При работе с массивами часто используют циклы (повторяющиеся вычисления). VBA предоставляет пользователям несколько вариантов организации циклов. В VBA существуют два основных типа циклов – *циклы со счетчиком* и *циклы с условием*. Циклы со счетчиком используют, когда необходимо выполнить тело цикла определенное количество раз, а циклы с условием – повторение в зависимости от некоторого постороннего условия.

1.1 Операторы цикла со счетчиком

Циклы со счетчиком еще называют циклами **«For»**, по имени оператора, с помощью которого он формируется:

For CounterVar=StartNum To EndNum [Step StepNum]
Операторы Тела Цикла
[Exit for]
Операторы Тела Цикла

Next [CounterVar]

где

- For – ключевое слово Visual Basic, обозначающее начало цикла;
- CounterVar – определенная пользователем переменная счетчика цикла;
- StartNum – начальное значение счетчика;
- To – ключевое слово Visual Basic, разделяющее «StartNum» и «EndNum»;
- EndNum – значение счетчика, после которого цикл завершается;
- Step – ключевое слово, используемое при указании шага цикла, необязательный аргумент;
- StepNum – значение шага цикла, т.е. значение, на которое увеличивается значение счетчика на каждом шаге. Это число может быть отрицательным;
- Next [CounterVar] – оператор, обозначающий конец цикла «For». Указывать в нем переменную счетчика не обязательно, хотя рекомендуется;
- Exit For – оператор принудительного завершения цикла.

Работа цикла происходит следующим образом:

- переменной цикла присваивается начальное значение ***StartNum***;

-выполняются ***Операторы Тела Цикла;***

-оператор ***Next*** возвращает управление оператору ***For***;

оператор ***For*** увеличивает значение переменной цикла на шаг и проверяет условие окончания цикла;

-если в теле цикла встречается оператор ***Exit For***, то происходит принудительный выход из текущего цикла (цикл прерывается и управление передается оператору следующим за оператором ***Next***)

При использовании вложенных циклов закрывается вначале внутренний, а затем внешний цикл. Если шаг цикла равен 1, то оператор ***Step*** опускают.

Пример 1 . Написать процедуру для нахождения суммы ряда:

$$\sum_{n=1}^{100} \frac{1}{n^2}$$

Текст программы приведен на рисунке 1.1.

```
Public Sub Сумма 100 ()
Dim n As Integer
‘ Переменная   n- целое число
Dim S As Single
‘ Переменная   S – вещественное число
S=0
‘Обнуление начальной суммы
‘Выполнение цикла
For n=1 To 100
   S=S+1/n^2
Next
‘Вывод результата в диалоговом окне
MsgBox(“Сумма 100 =   “ & S)
End Sub
```

Рис. 1.1 Процедура суммирования

1.2 Операторы цикла с условием Do … Loop

Циклом с условием является цикл **«Do…Loop»**. Такие циклы также называют *циклами с неизвестным числом повторений* или *итерационными*. Он продолжает свою работу в зависимости от состояния условия.

Главной особенностью таких циклов является наличие условия, т.е. любого логического выражения, функции или переменной, принимающих значения True (ИСТИНА) или False (ЛОЖЬ). Двумя основными циклами с условиями являются:

цикл **«Do…While»**, который выполняется *до тех пор, пока условие сохраняет значение True*, и

цикл **«Do…Until»**, работающий, *пока условие не станет равным True*.

Для вышеописанных циклов различают *циклы с пост-* и *предусловием.* Различаются они тем, где именно в цикле записывается условие, при этом принцип работы циклов не меняется.

Таблица 1

Синтаксис итерационных циклов

	Do…While	Do…Until
С предусловием	**Do While Условие** **Операторы** **Loop**	**Do Until Условие** **Операторы** **Loop**
С постусловием	**Do** **Операторы** **Loop While Условие**	**Do** **Операторы** **Loop Until Условие**

Единственным отличием циклов с пред- и постусловием является то, что в первом случае тело цикла может не выполнить ни одного раза, а во втором случае тело цикла всегда выполнится хотя бы один раз.

Пример 2 . Найти сумму ряда

$$\sum_{n=1}^{100} \frac{1}{n^2}$$

Суммирование продолжать до тех пор, пока сумма не достигнет значения 1,6.

Текст программы приведен на рисунке 1.2.

```
Public Sub СуммаРавнаЧислу()
Dim n As Integer
Dim S As Single
n=1
S=0
'Организация цикла – суммирование продолжать,
 пока сумма не достигнет значения 1,6
Do While S < 1.6
    S=S+1/(n^2)
    n=n+1
Loop
MsgBox("Сумма равна  " & S )
End Sub
```

Рисунок 1. 2 Процедура суммирования с условием

На рисунке 1.3 представлен код программы для решения Примера 2 с помощью оператора Do Until…Loop.

```
Public Sub СуммаРавнаЧислуUntil()
Dim n As Integer
Dim S As Single
n=1
S=0
'Организация цикла – суммирование продолжать,
'пока сумма не достигнет значения 1,6
Do Until S >= 1.6
S=S+1/(n^2)
n=n+1
Loop
MsgBox("Сумма равна  " & S )
End Sub
```

Рисунок 1. 3 Процедура суммирования с условием

Если возникает необходимость досрочного прерывания цикла **Do…Loop**, то необходимо использовать в необходимом месте тела цикла **Exit Do**. Как только компьютер встречает этот оператор, то цикл прерывается и управление передается оператору следующим за оператором **Loop**.

1.3 Оператор цикла с условием While… Wend

Оператор повторяет набор инструкций, пока условие выполняется.

```
While Условие=ИСТИНА
    Операторы тела цикла
Wend
```

На рисунке 1.4 представлен код программы для решения Примера 2 с помощью данного оператора.

```
Public Sub СуммаРавнаЧислу()
Dim n As Integer
Dim S As Single
n=1
S=0
'Организация цикла – суммирование продолжать,
 пока сумма не достигнет значения 1.6
While S < 1.6
      S=S+1/(n^2)
      n=n+1
Wend
MsgBox("Сумма равна    " & S )
End Sub
```

Рисунок 1. 4 Процедура суммирования с условием

1.4. Вложенные циклы

В некоторых случаях важно повторить подзадачу несколько раз внутри более общей задачи. Один из способов написания такой программы – включить в цикл набор инструкций (*внутренний цикл*), которые повторяются внутри другого цикла (*внешний цикл*). Такая структура, состоящая из цикла в цикле, называется *вложенными циклами*. Любой цикл может быть вложен в любой другой независимо от их видов. Например, можно вложить цикл **For** в такой же цикл **For** или в любой цикл **Do…Loop** и наоборот:

```
Do While condition
   For CounterVar = StartNum To EndNum
    …
   Next CounterVar
Loop
```

Особенно часто с вложенными циклами приходится сталкиваться при работе с многомерными массивами.

Варианты заданий

1. Массив A(10) расположен на листе ЭТ *Первый* в ячейках B1:B10. Вычислить значения массива C(10), каждый элемент которого получается возведением в куб элементов массива A. Разместить массив C(10) в ячейки B10:K10 листа с именем *Второй.*

2. Вычислить сумму элементов массива B(15). Вычисления следует прекратить, когда сумма достигнет величины 250. Исходные данные взять из диапазона ячеек B1:B15 листа *Исходные.* Сумму поместить в ячейку A2 листа *Данные*.
3. Массив N(4,5) расположен в ячейках C2:G5 листа ЭТ *Задание*. Найти произведение элементов каждого столбца массива и записать в массив P(5). Вывести окно с сообщением «Произведение j-го столбца равно P(j)».
4. Вычислить сумму элементов массива C(10). Вычисления следует прекратить, когда сумма достигнет величины 100. Исходные данные взять из диапазона ячеек B1:B10 листа *Массив.* Сумму поместить в ячейку A15 листа *Сумма*.
5. Массив M(3,4) расположен в ячейках A1:D3 листа ЭТ *Начало*. Найти сумму элементов каждой строки массива и записать в массив S (5). Вывести окно с сообщением «Сумма i-го столбца равно S(i)».
6. Массив M(3,4) расположен в ячейках A1:D3 листа ЭТ *Максимум*. Найти максимальный элемент. Вывести результат в диалоговом окне.
7. Вычислить сумму элементов массива C(20). Вычисления следует прекратить, когда сумма достигнет величины -350. Исходные данные взять из диапазона ячеек B1:B20 листа *Отрицательный.* Сумму поместить в ячейку A2 листа *Сумма*.
8. Вычислить количество отрицательных элементов массива C(20). Исходные данные взять из диапазона ячеек B1:B20 листа *Исходные*, результат вывести в диалоговом окне

9. Массив M(3,4) расположен в ячейках A1:D3 листа ЭТ *Минимум*. Найти минимальный элемент. Вывести результат в диалоговом окне.
10. Массив B(10) расположен в ячейках A1:A10 листа ЭТ *Начало*. Найти сумму отрицательных элементов. Вывести результат в диалоговом окне.
11. Массив M(5) расположен в ячейках A1:A5 листа ЭТ *Среднее*. Вычислить среднее арифметическое элементов массива и записать результат в ячейку B1 того же листа.
12. Массив A(8) расположен в ячейках C5:J5 листа ЭТ *Произведение.* Вычислить произведение всех элементов массива. Вывести результат в диалоговом окне.
13. Массив B(5,4) расположен в ячейках C4:F8 листа ЭТ *Массив*. Найти максимальный и минимальный элементы. Записать результаты в ячейки A1 и A2 того же листа.
14. Массив A(10) расположен в ячейках B14:K1410 листа ЭТ *Массив*. Найти сумму положительных элементов. Вывести результат в диалоговом окне.
15. Массивы A(7) и B (7) расположены в ячейках E3:K3 и E6:K6 листа ЭТ *Данные*. Получить массив C97) путём сложения соответствующих элементов массивов A и B. Результат записать в ячейку D3:D9 листа *Результат.*

2. Работа с подпрограммами

Как и все алгоритмические языки высокого уровня Visual Basic позволяет осуществлять обращение из одной

процедуры в другую, а так же передавать параметры (аргументы) из одной процедуры в другую.

Обращения из одной процедуры в другую осуществляется посредством указания имени процедуры, к которой происходит обращение. В основной программе передаваемые параметры называются *Фактическими*, а в подпрограмме –*Формальными.*

Задание *Формальных* параметров в подпрограмме выглядит следующим образом:

Public Sub Имя процедуры (формальные параметры)

А обращение из основной программы имеет вид:

Call Имя процедуры (фактические параметры)

Пример 1. Вычислить сумму элементов двух матриц A(3,3) и B (3,3). Для ввода, вывода и суммирования матриц использовать отдельные процедуры.

Процедура для ввода матриц изображена на рисунке 2.1. В данной процедуре используются следующие формальные параметры Sheet –номер листа, row-номер строки, col-номер столбца, e()- имя вводимой матрицы , n, m –целые числа для работы цикла. Запись row + i - 1, col + j – 1 является универсальной, и определяет адрес ячейки для любого случая.

```
Public Sub vvod(Sheet As Integer, row As Integer, col As
Integer, e() As Single, n As Integer, m As Integer)
Dim I As Integer, j As Integer
For i = 1 To n
   For j = 1 To m
      e(i, j) = Worksheets(Sheet).Cells(row + i - 1, col + j      -
1).Value
   Next j
Next i
```

```
End Sub
```

Рисунок 2.1 Процедура для ввода матриц

Процедура для суммирования изображена на рисунке 2.2.

```
  Public Sub summa(S() As Single, n As Integer, m As
Integer, P() As Single, L() As Single)
  Dim i As Integer, j As Integer
  For i = 1 To n
     For j = 1 To m
        S(i, j) = P(i, j) + L(i, j)
     Next j
  Next i
```

Рисунок 2.2 Процедура для сумирования матриц

Процедура вывода результирующей матрицы представлена на рисунке 2.3. Здесь переменная *nam* это текстовый комментарий, который вставляется в соответствующую ячейку.

```
    Public Sub vivod(Sheet As Integer, row As Integer,
col As Integer, n As Integer, m As Integer, z() As Single,
nam As String)
    Dim i As Integer, j As Integer
    Worksheets(Sheet).Cells(row -1, col).Value = nam
    For i = 1 To n
       For j = 1 To m
       Worksheets(Sheet).Cells(row + i - 1, col + j -
1).Value =    z(i, j)
       Next j
    Next i
    End Sub
```

Рисунок 2.3 Процедура для сумирования матриц

Основная процедура представлена на рисунке 2.4. В скобках указаны фактические параметры, которые принимают значения соответствующих переменных, объявленных в описании данной процедуры.

```
Public Sub matrix()
Dim A(3, 3) As Single
Dim B(3, 3) As Single
Dim C(3, 3) As Single
        ‘ Обращение к процедуре ввода матриц.
Call vvod(1, 2, 1, A(), 3, 3)
        ‘ Обращение к процедуре ввода матриц.
Call vvod(1, 2, 5, B(), 3, 3)
‘ Обращение к процедуре суммирования матриц Call
summa(C(), 3, 3, A(), B())
‘ Обращение к процедуре вывода матрицы.
Call vivod(1, 7, 1, 3, 3, C(), "СУММ")
End Sub
```

Рисунок 2.4 Основная прроцедура

Пример 2. Выполнить транспонирование матрицы A(3,3). На рисунке 2.5 и 2.6 представлен листинг программы для транспонирования заданной матрицы.

```
Public Sub matrix()
Dim A(3, 3) As Single
Dim B(3, 3) As Single
Call vvod(2, 1, 1, A(), 3, 3)
Call transp(A(), 3, 3, B())
Call vivod(2, 5, 1, 3, 3, B(), "Трансп")
End Sub
```

Рисунок 2.4 Основная процедура для транспонирования матрицы

```
Public Sub vvod(Sheet As Integer, row As Integer, col As
Integer, e() As Single, n As Integer, m As Integer)
Dim i As Integer, j As Integer
For i = 1 To n
For j = 1 To m
 e(i, j) = Worksheets(Sheet).Cells(row + i - 1, col + j - 1).Value
     Next j
       Next i
End Sub

Public Sub transp(M1() As Single, n As Integer, m As
Integer, M2() As Single)
Dim i, j As Integer
For i = 1 To n
 For j = 1 To m
 M2(i, j) = M1(j, i)
        Next j
       Next i
End Sub

Public Sub vivod(Sheet As Integer, row As Integer, col As
Integer, n As Integer, m As Integer, z() As Single, nam As
String)
Dim i, j As Integer
Worksheets(Sheet).Cells(row -1, col).Value = nam
For i = 1 To n
For j = 1 To m
  Worksheets(Sheet).Cells(row + i - 1, col + j - 1).Value = z(i,
j)
     Next j
       Next i
End Sub
```

Рисунок 2.6 Подпрграммы для ввода, выводла
и транспонирования матрицы

Пример 3. Выполнить матричное умножение матриц A(4,3) и B(3,2). На рисунке 2.7 представлен листинг программы для транспонирования заданной матрицы.

```
Public Sub matrix()
Dim A(4, 3) As Single
Dim B(3, 2) As Single
Dim C(4, 2) As Single
Call vvod(3, 1, 2, A(), 4, 3)
Call vvod(3, 5, 2, B(), 3, 2)
Call mumnog(A(), B(), C(), 4, 2, 3)
Call vivod(3, 9, 1, 4, 2, C(), "Произв")
End Sub
Public Sub vvod(Sheet As Integer, row As Integer, col As
Integer, e() As Single, n As Integer, m As Integer)
Dim i, j As Integer
For i = 1 To n
   For j = 1 To m
 e(i, j) = Worksheets(Sheet).Cells(row + i - 1, col + j   -
1).Value
  Next j
    Next i
End Sub
Public Sub mumnog(K1() As Single, K2()  As Single, K3()
As Single , n As Integer, m As Integer, l As Integer)
Dim i As Integer, j As Integer, r As Integer
For i = 1 To n
   For j = 1 To m
      K3(i, j) = 0
      For r = 1 To l
  K3(i, j) = K3(i, j) + K1(i, r) * K2(r, j)
    Next r
     Next j
      Next i
End Sub
```

Рисунок 2.7 Листинг процедуры
для выполнения матричного умножения

```
  Public Sub vivod(Sheet As Integer, row As Integer, col
As Integer, n As Integer, m As Integer, z() As Single,
nam As String)
  Dim i As Integer, j As Integer
  Worksheets(Sheet).Cells(row + i, col).Value = nam
  For i = 1 To n
     For j = 1 To m
        Worksheets(Sheet).Cells(row + i - 1, col + j -
1).Value = z(i, j)
     Next j
  Next i
  End Sub
```

Рисунок 2.8 Продолжение листинга процедуры для выполнения матричного умножения

Пример 4. Умножить матрицу A(3,3) на число 5. Листинг программы представлен на рисунках 2.9 и 2.10.

```
    Public Sub vvod(Sheet As Integer, row As Integer, col As
Integer, e() As Single, n As Integer, m As Integer)
    Dim i As Integer , j As Integer
    For i = 1 To n
       For j = 1 To m
          e(i, j) = Worksheets(Sheet).Cells(row + i - 1, col + j -
1).Value
       Next j
    Next i
    End Sub

    Public Sub umnogch(T1() As Single, n As Integer, m As
Integer, ch As Single, T2() As Single)
    Dim i As Integer, j As Integer
    For i = 1 To n
       For j = 1 To m
          T2(i, j) = ch * T1(i, j)
       Next j
```

Рисунок 2.9 Листинг программы умножения матрицы на число

```
Next i
End Sub

Public Sub vivod(Sheet As Integer, row As Integer, col As
Integer, n As Integer, m As Integer, z() As Single, nam As
String)
Dim i, j As Integer
Worksheets(Sheet).Cells(row + i, col).Value = nam
For i = 1 To n
   For j = 1 To m
      Worksheets(Sheet).Cells(row + i - 1, col + j - 1).Value =
z(i, j)
   Next j
Next i
End Sub

Public Sub chislo()
Dim A(3, 3) As Single
Dim B(3, 3) As Single
Call vvod(1, 1, 1, A(), 3, 3)
Call umnogch(A(), 3, 3, 5, B())
Call vivod(1, 7, 1, 3, 3, B(), "Умнож")
End Sub
```

Рисунок 2.10 Продолжение листинга программы умножения матрицы на число

Варианты заданий

Написать программу для вычисления матричноого выражения. Все действия выполнить последовательно в одной основной процедуре с обращениями к подпрограммам.

Таблица 2

№	Матричное выражение	№	Матричное выражение
1	$((Q^T_{34}+D_{43})H_{32})^T=?$	**9**	$((Q^T_{34}-D_{43})H_{32})^T=?$
2	$(B^T_{23}+H_{32})(E_{22}+D_{22})=?$	**10**	$(B^T_{23}-H_{32})(E_{22}+D_{22})=?$
3	$(Q^T_{34}D_{34}+E_{44})^T=?$	**11**	$(Q^T_{34}D_{34}+E_{44})^T=?$
4	$(E_{33}+H_{33}+D^T_{33})Q_{34}=?$	**12**	$(E_{33}+H_{33}-D^T_{33})Q_{34}=?$
5	$((E_{44}+D^T_{44})Q_{43}-B_{43})^T=?$	**13**	$((E_{44}+D^T_{44})Q_{43}+B_{43})^T=?$
6	$((H_{34}B_{43})^T+E_{33}-D_{33})^T=?$	**14**	$D_{43}(E_{33}+H_{33})^T+Q^T_{34}=?$
7	$((D_{34}+B_{34})Q_{43})^T+E_{33}=?$	**15**	$(D_{33}+E_{33})^T+H_{34}Q_{43}=?$
8	$(D^T_{34}(E_{33}+B_{33}+H_{33}))^T=?$	**16**	$(Q_{34}B^T_{34}+E_{33}-D_{33})^T=?$

3. Работа с диаграммами

3.1 Построение диаграммы средствами VBA

Чтобы построить график средствами VBA, необходимо:

1. Указать диапазоны значений функций, рассчитанных ранее и находящиеся на листе MS Excel.
2. Построить простейшую диаграмму для выбранных значений. Простейшая – в том смысле, что для ее создания используется минимум информации для форматирования будущей диаграммы.
3. Изменить тип диаграммы, который был создан по умолчанию, на необходимый.
4. Изменить значения по оси x на значения аргумента функции. Пока там указаны номера строк.
5. Создать заголовок диаграммы.
6. Подписать оси координат диаграммы, сначала сделав их видимыми.
7. Записать в легенду комментарии к созданным графикам.

Последовательность действий необходимо соблюдать для первых трех пунктов, далее действия могут выполняться в произвольном порядке, и даже могут быть пропущены.

Пример1. Построить график функции *y(x)=sin(x)* на интервале значений аргумента от *0* до *6,4* с шагом *0,2*.

1. Табулируем значения аргумента с заданными параметрами в диапазоне ячеек A1:A33 и рассчитаем по ним значения функции по адресам B1:B33 (рис3.1).

B1 =SIN(A1)

	A	B	C	D	E
1	0	0			
2	0,2	0,198669			
3	0,4	0,389418			
4	0,6	0,564642			
5	0,8	0,717356			
6	1	0,841471			
32	6,2	-0,08309			
33	6,4	0,116549			

Рисунок 3..1 Исходные данные для построения графика

2. Строим простейшую диаграмму на текущем листе MS Excel (*ActiveSheet*) достаточно в коллекцию фигур (*Shapes*) текущего листа добавить новую диаграмму (*AddChart*). После указания ключевого слова *Select* она становится активной, или текущей диаграммой (*ActiveChart*). Для нее необходимо указать адрес, где находятся исходные данные (рис.3. 2). Свойства новой диаграммы, такие, как тип диаграммы, цвет и толщина линий, заголовок, подписи по осям, легенда и другие будут установлены по умолчанию.

cbGo

```
Private Sub cbGo_Click()
    'На активный лист к графичеким объектам добавляется диаграмма
    ActiveSheet.Shapes.AddChart.Select
    'Задает диапазон исходных данных графика
    ActiveChart.SetSourceData Source:=Range("Лист1!b1:b33")
End Sub
```

Рисунок 3. 2 Простейшая программа для построения графика

3. Определяем источник данных для диаграммы посредством метода *SetSourceData* при помощи ключевого

слова *Source*. По умолчанию, данные должны быть расположены в столбце. Если данные расположены в строке, после указания источника данных через запятую необходимо использовать ключевое слово *PlotBy* с параметром *xlRows*, например, *PlotBy:=xlRows*. Чтобы построить два графика в одних координатах, в качестве источника данных необходимо указать два столбца исходных данных, например: *A1:B33*.

4. Результат выполнения программы представлен на рисунке3.3. Поведение функции на нем уже заметно, но, прежде чем использовать диаграмму на экранной форме или текстовом документе, необходимо выполнить её форматирование.

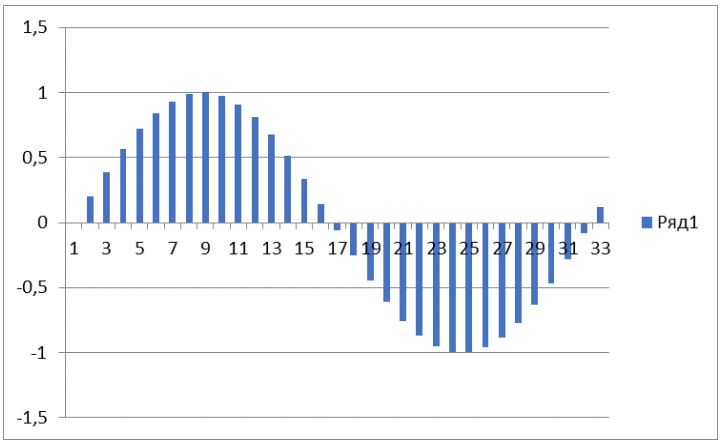

Рис.3.3 Простейший график функции

5. Для текущей (активной) диаграммы *ActiveChart*, для ее свойства *ChartType* выбираем нужный тип диаграммы из предусмотренных в *VBA* (рис3.4).

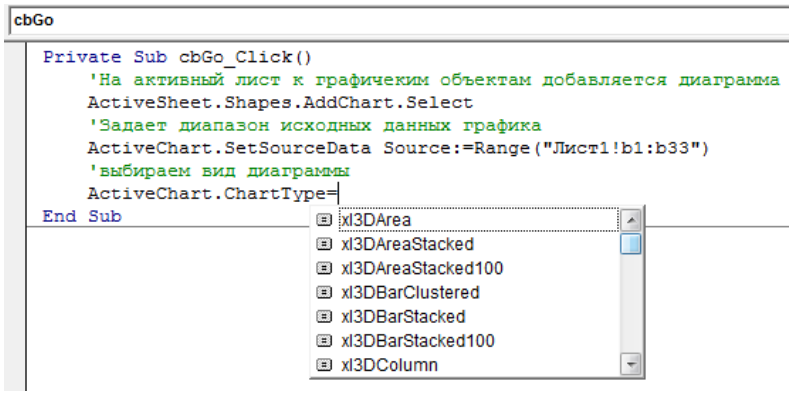

Рисунок 3.4 Изменение типа диаграммы

6. Выбираем . диаграмму типа – *xlXYScatterSmoothNoMarkers* – двухмерная линейная гладкая диаграмма без маркеров в декартовых координатах (рис.3.5).

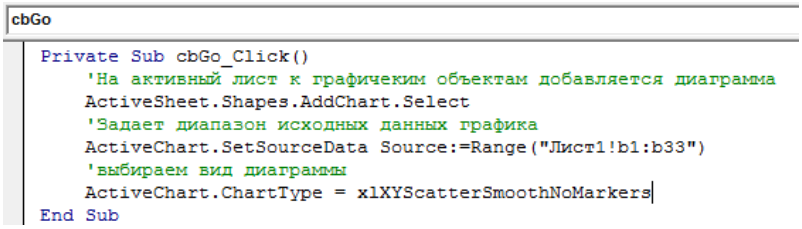

Рисунок 3.. 5 Изменение типа диаграммы на xlXYScatterSmoothNoMarkers

Как видно из рисунка 3.6, это один из наиболее распространенных типов диаграмм. Он сглаживает (интерполирует) функцию между точками.

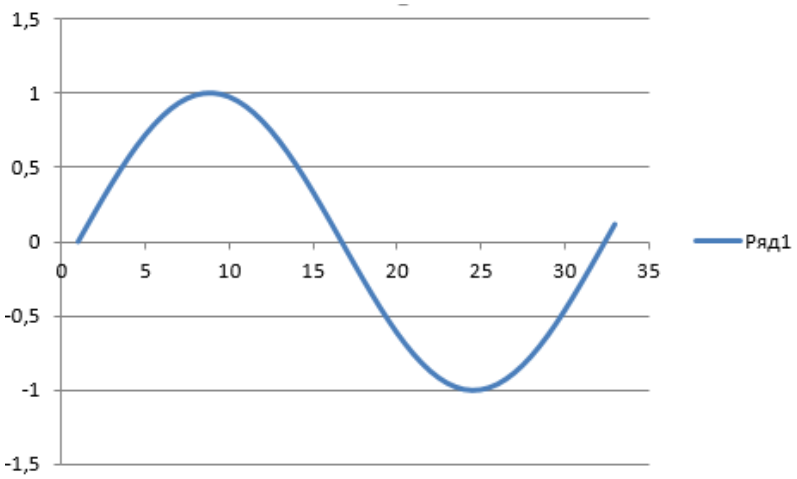

Рисунок 3..6 Вид диаграммы со значением ActiveChart.ChartType = xlXYScatterSmoothNoMarkers

В некоторых случаях предпочтительно использовать тип диаграмм с закрашенными областями (рис.3.7). Закрашенные области чаще всего встречаются в линейчатых и столбиковых диаграммах.

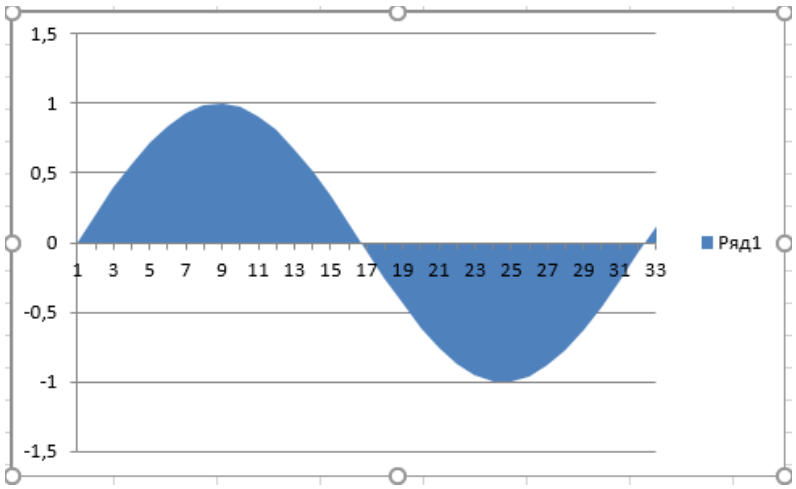

Рисунок 3..7 Вид диаграммы со значением ActiveChart.ChartType = xlArea

Иногда целесообразно использовать художественные типы диаграмм, например, пузырьковые (рис.3.8). Об инженерном использовании диаграммы в этом случае говорить не приходится, а в декоративных целях этот тип диаграмм имеет право на существование.

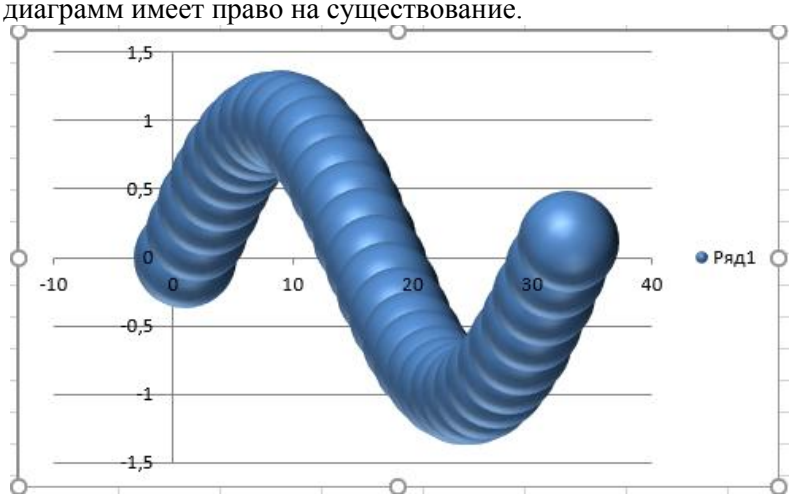

Рисунок 3..8 Вид диаграммы со значением ActiveChart.ChartType = xlBubble3DEffect

Некоторые типы диаграмм позволяют даже поворачивать область диаграммы (рис3.9). Этим свойством обладают линейчатые диаграммы – они напоминают столбчатые диаграммы, развернутые на 90 градусов.

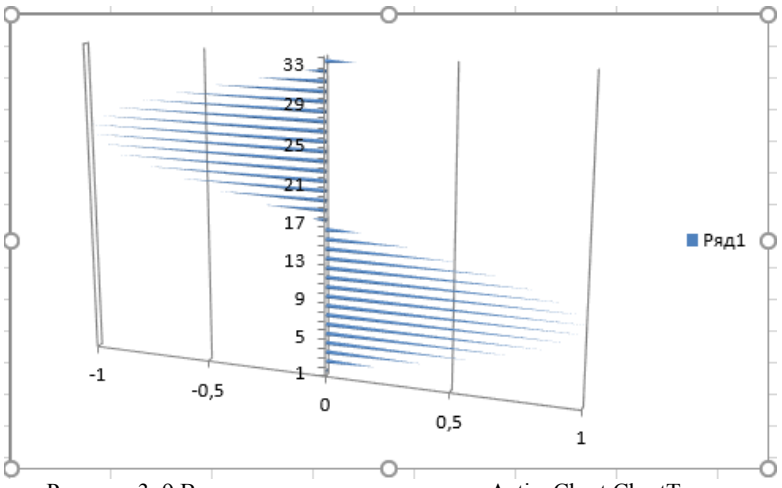

Рисунок 3..9 Вид диаграммы со значением ActiveChart.ChartType = xlConeBarStacked

Можно изменить цвет линий на диаграмме. Для этого используется свойство *ChartColor* объекта *ActiveChart*. Цвет указывается, например, десятичным числом от 1 до 26.

ActiveChart.ChartColor = 17

7. Отображаем подписи по осям диаграммы, для этого сначала нужно сделать поля, где будут находиться подписи, видимыми. Можно сделать видимыми подписи по обеим осям координат, или по любой одной. Для активной диаграммы все операции с осями координат объединены в методе *Axes*. Ось *X* обозначена как *xlCategory*, а ось *Y* – *xlValue*. Свойство, определяющее видимость подписей по оси, называется *HasTitle*. Его надо установить в *True* (или равным единице).

8. Указываем актуальные значения диапазона x на оси абсцисс. На рисунах 3. 8 и 3. 9 то заметно, что по оси абсцисс пока отложены номера строк в таблице исходных данных.. Для этого диапазон значений x записывается в свойство *XValues* (значения переменной *X*).

ActiveChart.FullSeriesCollection(1).XValues = ("Лист1!a1:a33")

Метод *FullSeriesCollection* указывает, что значения *x* будут одинаковы для всех линий (функций) диаграммы - объектов *SeriesCollection*. Данную строку программного кода можно вставить в произвольное место, но рекомендуется расположить ее рядом с указанием исходных данных для построения диаграммы (рис.3.10).

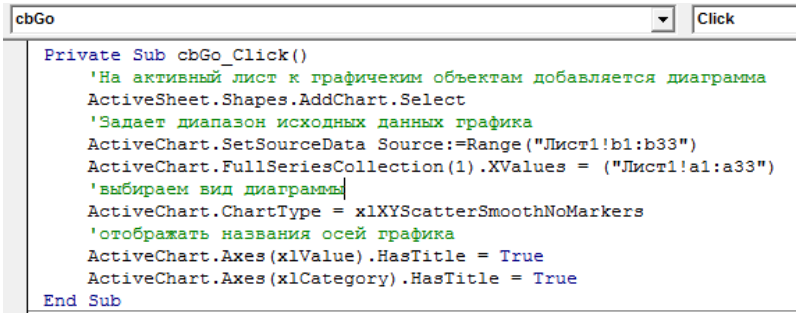

Рисунок 3. .10 Задание значений по оси x и видимости названий осей диаграммы

На рисунке.3.11. Значения по оси x соответствуют заданному диапазону. Подписи по осям видимы, но они не несут содержательной информации.

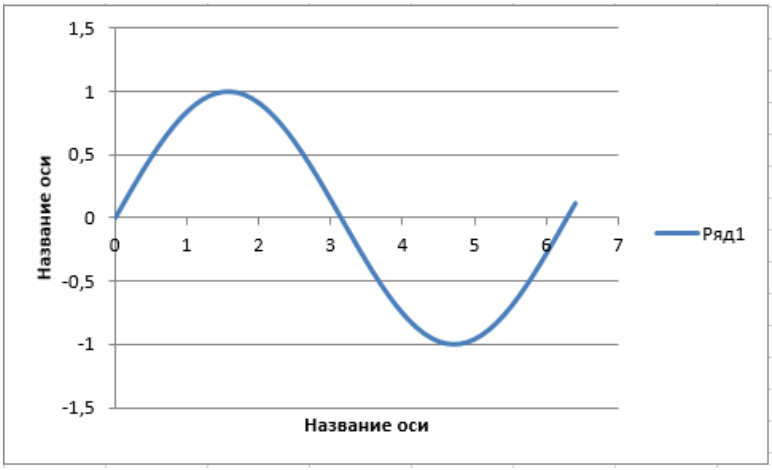

Рисунок .3. 11 Вид диаграммы со значениями по оси x и видимыми названиями осей

9. Изменяем название осей. С помощью объекта *AxisTitle*. Он входит в более высокий по иерархии объект *Axes*. У *AxisTitle* есть различные свойства, определяющие текст, шрифт, его размер, цвет, выравнивание текста. Н для изменения названия осей ужно записать в свойство *Text* обеих осей их названия. Так как тип свойства – *string*, то названия осей должны быть в двойных кавычках (рис.3. 12). Такой же результат получим, если вместо свойства *Text* использовать свойство *Caption*. Управлять шрифтом подписей е названиям осей можно при помощи свойств *Font.Name* и *Font.Size*.

```
cbGo                                    Click
Private Sub cbGo_Click()
    'На активный лист к графичеким объектам добавляется диаграмма
    ActiveSheet.Shapes.AddChart.Select
    'Задает диапазон исходных данных графика
    ActiveChart.SetSourceData Source:=Range("Лист1!b1:b33")
    'выбираем вид диаграммы
    ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
    'отображать названия осей графика
    ActiveChart.Axes(xlValue).HasTitle = True
    ActiveChart.Axes(xlCategory).HasTitle = True
    'ввод названия осей графика
    ActiveChart.Axes(xlValue).AxisTitle.Select
    ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Text = "Ось Y"
    ActiveChart.Axes(xlCategory).AxisTitle.Select
    ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Text = "Ось X"
End Sub
```

Рисунок 3. .12 Изменение названий осей координат диаграммы

На рисунке 3.13 подписи по осям диаграммы приведены в соответствие.

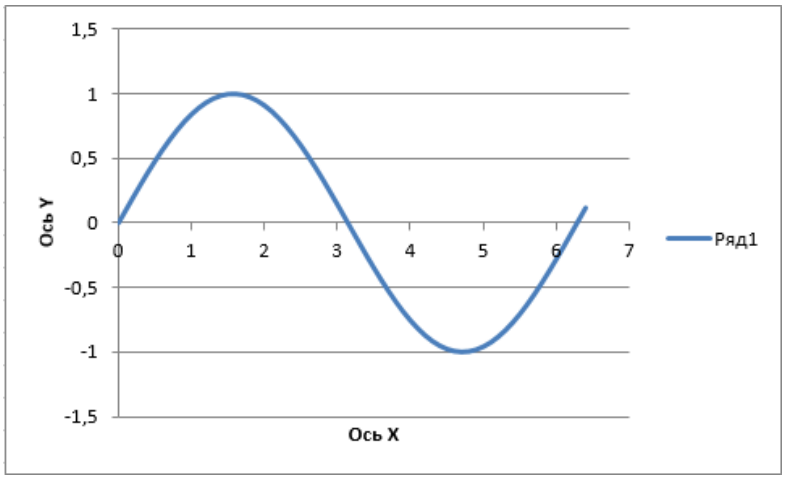

Рисунок 3. .13 Вид диаграммы с подписями по осям

10. Создаем заголовок диаграммы. Для объекта активной диаграммы *ActiveChart* существует свойство *HasTitle*, определяющее видимость поля заголовка диаграммы. По аналогии с полями подписей к осям диаграммы, его тоже необходимо сделать сначала видимым (рис.3.14). Поле

заголовка диаграммы – это объект *ChartTitle*. Заголовок следует записать в свойство *Text* или *Caption*.

```
cbGo                                                    Click
    'ввод названия осей графика
    ActiveChart.Axes(xlValue).AxisTitle.Select
    ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Text = "Ось Y"
    ActiveChart.Axes(xlCategory).AxisTitle.Select
    ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Text = "Ось X"
    'Название графика
    ActiveChart.HasTitle = True
    ActiveChart.ChartTitle.Text = "Пример графиков"
End Sub
```

Рисунок 3..14 Изменение заголовка диаграммы

На рисунке 3. 15 заголовок присутствует. Для него также возможно изменить шрифт, его размер, начертание, цвет, выравнивание и его положение на диаграмме при помощи соответствующих свойств.

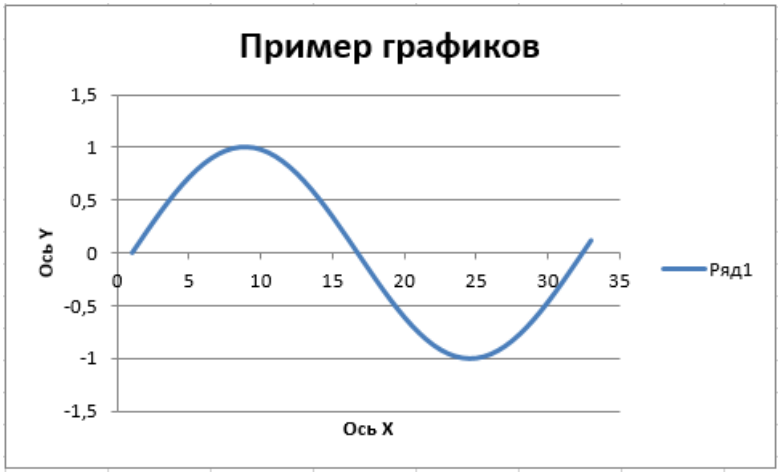

Рисунок 3..15 Вид диаграммы с измененным заголовком

11. Изменяем подписи в легенде. Поскольку в диаграмме по умолчанию легенда уже видима, то этот этап здесь пропущен. Для обращения к первой функции, второй и

т.д. следует использовать их порядковый индекс в коллекции подобных объектов. Это позволяет сделать объект *SeriesCollection*. Подпись к функции в легенду записывается в свойство *Name* (рис.3. 16). Свойства *Caption* или *Text* здесь не поддерживается.

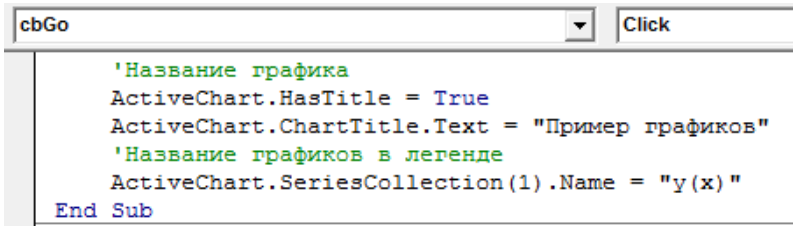

Рисунок 3. .16 Изменение легенды диаграммы

Если наоборот, легенду нужно убрать (например, если на графике только одна функция, то легенда не очень-то и нужна), то следует свойство *HasLegend* объекта *ActiveChart* сделать ложным:

ActiveChart.HasLegend = False

Диаграмма с измененной легендой представлена на рисунке 3. 17. Здесь также можно изменить начертание и размер шрифта, положение легенды на диаграмме.

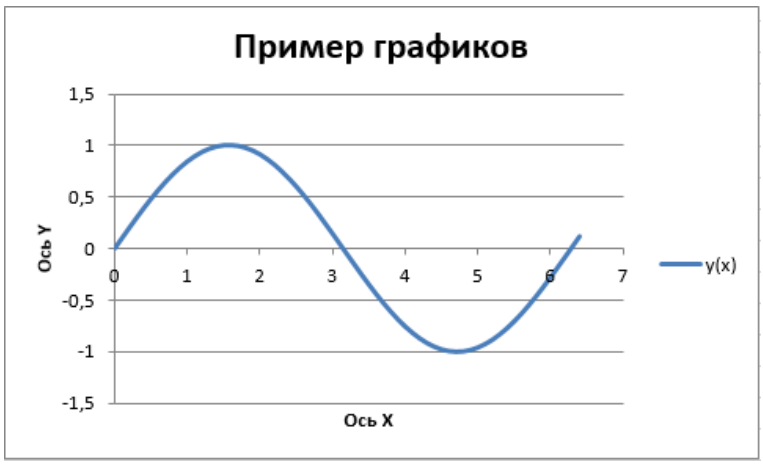

Рисунок 3..17 Окончательный вид диаграммы, построенной средствами VBA

12. Выводим полученный график на экранную форму VBA. Эта цель достигается за два шага. Первый шаг заключается в экспорте полученной диаграммы в графический файл.

```
ActiveChart.Export "Graphic.bmp"
```

При помощи данной строки активная диаграмма сохраняется в файл с указанным именем и расширением. Формат файла не зависит от расширения, которое будет здесь указано. По умолчанию, это пиксельная (растровая) матрица цветов – *bmp*. В большинстве случаев, данный формат устраивает. В случаях, когда его невозможно использовать, формат графики можно изменить при помощи необязательного ключевого параметра *FilterName*. Если путь к файлу не указан, то он сохраняется в том же месте, где находится открытый «родительский» файл MS Excel. При многократной записи диаграммы в один и тот же файл его содержимое каждый раз будет переписываться без предупреждения.

На втором шаге диаграмма из файла вставляется в заранее подготовленный объект *Image*, и выполняется необходимое его форматирование

```
imgGraph.Picture = LoadPicture("Graphic.bmp")
```

Здесь *imgGraph* – имя объекта типа *Image*. Очевидно, в других программах можно дать этому объекту другое имя. Важно, чтобы данный объект с именем уже существовал на форме. В противном случае при выполнении программы получим сообщение об ошибке, например:

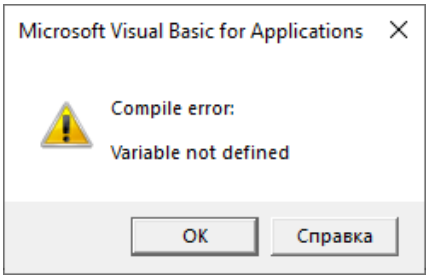

Рис 3. 18 Сообщение об ошибке при неправильном имени объекта

Текст сообщения может быть иным, например: *объекта с таким именем не существует*. Во всех случаях следует тщательно сравнить имя объекта в программе со свойством *Name* в *Properties Explorer*.

Если рисунок отображается, то следует воспользоваться свойствами и методами объекта *Image* для правильного представления диаграммы. Использование свойств и методов может быть сделано на этапе разработки или выполнения программы. Некоторые из свойств и методов приводится здесь:

AutoSize – автоматическое масштабирование диаграммы по размеру объекта *Image*.

PictureAlignment – выравнивание диаграммы в окне *Image*. Возможные варианты: по верхнему, левому, нижнему, правому краю или по центру.

Visible – указывает, видим ли объект *Image*, или нет.

Конечный результат работы программы представлен на рисунке 3.19:

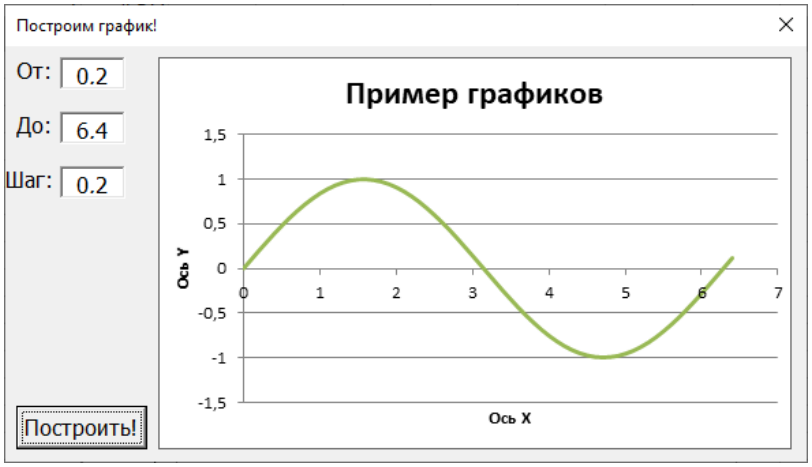

Рисунок 3. .19 Результат работы программы

13 Усовершенствуем полученную программу следующим образом

- На листе Excel и на диске остались диаграммы. Сколько раз запускается программа, столько графиков остается на листе Excel. Поэтому целесообразно каждый раз после сохранения диаграммы в файле удалять ее с листа Excel

ActiveChart.Parent.Delete

Графический файл при каждом запуске программы переписывается, поэтому в размере он не увеличивается. При желании его тоже можно удалить:

Kill “Graphic.bmp”

- Исходные данные для построения диаграммы уже не нужны и их тоже можно удалить. Чтобы этот лист Excel не казался лишним и его можно было использовать, рекомендуется исходные данные записывать в ячейки с номерами более 400 для строк и столбцов. Далеко не на каждом листе используются ячейки с такими номерами.

3.2 Построение диаграмм средствами Макрорекордера

Описанный выше способ не единственный для построения диаграмм средствами VBA Более того, даже при точном копировании всех вышеописанных действий в процессе выполнения программы могут обнаружится ошибки. Причиной их в большинстве случаев будет различие версий Microsoft Office. Чтобы быстро создать работающую программу, причем не только для построения графиков, рекомендуется использовать Макрорекордер - приложение, формирующее макрос – приложение Visual Basic for Application из тех действий, которые выполняются при помощи мыши и клавиатуры.

Запись макросов осуществляется путем выбора пункта «Запись макроса» в группе «Код» на вкладке ленты «Разработчик» (рис.3.20).

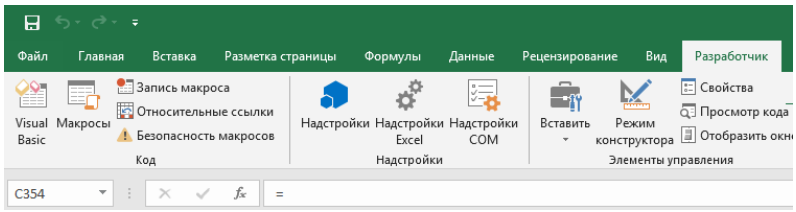

Рисунок 3. .20 Место кнопки записи макроса

После нажатия кнопки появится окно, в котором указывается имя макроса, место его сохранения – по

умолчанию, это открытый файл MS Excel (Эта книга). Кроме этого, необходимо назначить сочетание клавиш, при нажатии которого и начнется выполнение макроса. Текст в описании помогает выбрать нужный макрос, когда их много, а названия недостаточно, чтобы понять назначение макроса (рис3. 21).

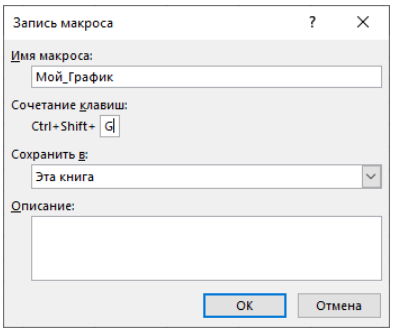

Рисунок 3..21 Окно свойств записи макроса

Сразу после закрытия окна начинается запись всех действий в книге Excel в макрос с выбранным именем. Адреса ячеек (диапазонов) без преобразований включается в текст макроса, выбранные пункты меню заменяются командами VBA. Это продолжается до тех пор, пока не выбран пункт «Остановить запись» (рис.3. 22):

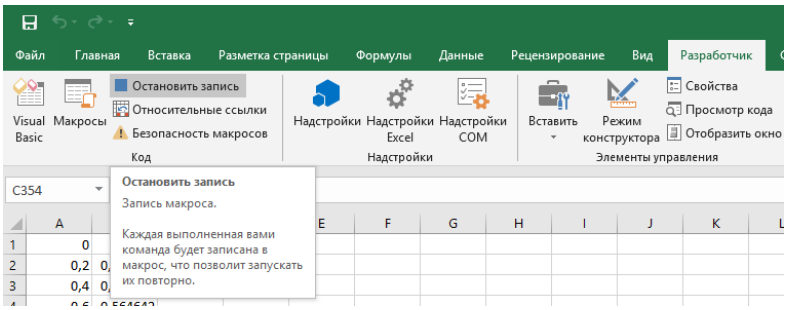

Рисунок 3. .22 Место кнопки остановки записи макроса

В результате получается подпрограмма на VBA, записанная в отдельный модуль. Ее можно посмотреть и отредактировать, открыв среду разработки.

Достоинством такого способа является отсутствие ошибок при выполнении полученной программы. Недостатком – нужно точно представлять последовательность действий в Excel для получения конечного результата. В противном случае неправильные действия тоже будут включены в макрос.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Написать программу в среде Visual Basic for Application, реализующую:

- Вычисление значений функции Z(x).
- Ввод границ интервала, шага осуществить в экранной форме.
- Построить график функции на экранной форме, со всеми необходимыми подписями по осям, названию графика и легенды.
- Удалить с листа MS Excel вспомогательные расчеты и построения.

Таблица 3

№	Уравнение	**a**	**b**	h
1.	$z(x)=\begin{cases}\dfrac{1+\lvert x\rvert}{\sqrt[3]{1+x+x^2}}, & при\ x\le -1\\ 2\cdot\ln\left(1+x^2\right)+\dfrac{1+\cos^4(x)}{2+x}, & при\ -1<x<0\\ \sqrt[5]{(1+x)^3}, & при\ x\ge 0\end{cases}$	-3	2	0,1

№	Уравнение	**a**	**b**	h
2.	$z(x)=\begin{cases}3\cdot x+\sqrt{1+x^2}, & при\ x<0\\ 2\cdot\cos(x)\cdot e^{-2\cdot x}, & при\ \ 0\le x\le 1\\ 2\cdot\sin(3\cdot x), & при\ \ x>1\end{cases}$	-2	3	0,1
3.	$z(x)=\begin{cases}\dfrac{1+x+x^2}{1+x^2}, & при\ x<0\\ \sqrt{1+\dfrac{5\cdot x}{1+x^3}}, & при\ \ 0\le x<1\\ 5\cdot\lvert 0,7\cdot\cos(x)+\sin(x)\rvert, & при\ \ x\ge 1\end{cases}$	-2	3	0,1
4.	$z(x)=\begin{cases}\sqrt{1+\dfrac{x^2}{1+x^4}}, & при\ x<0\\ 2\cdot\sin^3(x), & при\ \ 0\le x\le 1\\ \sqrt{1+\sqrt[3]{\lvert 2\cdot\cos(6\cdot x)\rvert}}, & при\ \ x>1\end{cases}$	-2	3	0,1
5.	$z(x)=\begin{cases}\dfrac{1+x^2}{\sqrt{1+x^4}}, & при\ x<0\\ 2\cdot x+\dfrac{\sin^2(x)}{2+x}, & при\ \ 0\le x\le 1\\ \sqrt{1+\sqrt[3]{\lvert 2\cdot\cos(6\cdot x)\rvert}}, & при\ \ x>1\end{cases}$	-2	2	0,1
6.				

7.	$z(x)=\begin{cases}\sqrt{1+\dfrac{x^2}{1+x^4}}, & npu\ x<0\\ \dfrac{5\cdot x^2}{1+x^2}, & npu\ \ 0\le x\le 1\\ \sqrt{1+\dfrac{2\cdot x}{1+x^2}}, & npu\ \ x>1\end{cases}$	-2	2	0,1
8.	$z(x)=\begin{cases}\dfrac{\sqrt{1+\lvert x\rvert}}{2+\lvert x\rvert}, & npu\ x\le -1\\ 2\cdot\ln\left(1+x^2\right)+\dfrac{1+\cos^4(x)}{2+x}, & npu\ \ -1<x<0\\ \dfrac{1+x}{2+\cos^3(x)}, & npu\ \ x\ge 0\end{cases}$	-2	2	0,1
9.	$z(x)=\begin{cases}\dfrac{1+5\cdot x}{3+x^2}, & npu\ x<0\\ \sin^3(x)\cdot\sqrt{5+x}, & npu\ \ 0\le x<1\\ \sin^3(x+1)\cdot e^{0,6\cdot x}, & npu\ \ x\ge 1\end{cases}$	-2	3	0,1
10.	$z(x)=\begin{cases}\sqrt{1+x^2}, & npu\ x\le 0\\ \dfrac{1+x^3}{1+\sqrt[5]{1+e^{-\frac{x}{2}}}}, & npu\ \ 0<x<1\\ \sqrt[5]{(1+x)^3}, & npu\ \ x\ge 0\end{cases}$	-2	2	0,1

11.	$z(x)=\begin{cases}\dfrac{1+\lvert x\rvert}{\sqrt[3]{1+x+x^2}}, & npu\ x\le -1\\ \sqrt{1+\dfrac{5\cdot x}{1+x^3}}, & npu\ -1<x<0\\ \sqrt[5]{(1+x)^3}, & npu\ x\ge 0\end{cases}$	-2	3	0,1
12.	$z(x)=\begin{cases}3\cdot x+\sqrt{1+x^2}, & npu\ x<0\\ 2\cdot\cos(x)\cdot e^{-2\cdot x}, & npu\ 0\le x\le 1\\ 2\cdot\sin(3\cdot x), & npu\ x>1\end{cases}$	-2	3	0,1
13.	$z(x)=\begin{cases}\dfrac{\lvert x\rvert}{1+x^2}\cdot e^{-5\cdot x}, x<0\\ \sqrt{1+x^4}, 0\le x<1\\ \dfrac{1+\cos(\pi\cdot x)}{6+x}+3\cdot x, x\ge 1\end{cases}$	-1,5	1,5	0,1
14.	$z=\begin{cases}2\cdot\sqrt[3]{x}+\sqrt[3]{\lvert x\rvert^2}, x<-1\\ \dfrac{1}{3+x+\ln(\lvert x+1\rvert)}, -1\le x\le 0,5\\ \lvert x^2-4\rvert, x>0,5\end{cases}$	-1,5	1,5	0,1
15.	$z=\begin{cases}\sqrt[3]{x^2-1}, x<-0,5\\ \sin^3(\pi\cdot x)\cdot\dfrac{2+x}{1+\cos^2(x)}, & -0,5\le x\le 0,5\\ \sin(3\cdot x), & x<0,5\end{cases}$			

Библиографический список

.

1. *Уокенбах Д.* MS Excel 2010: профессиональное программирование в VBA.- М.: Вильямс, 2012. - 944 с.
2. Фризен, И. Г. Офисное программирование : учебное пособие. - Ростов-на-Дону: Феникс, 2010. - 241 с.
3. *Векшина Н.В.* Информатика. Прикладное программирование на языке VISUAL BASIC. Методические указания к лабораторным работам и самостоятельной работе для студентов всех направлений.- СПб: Национальный минерально-сырьевой университет «Горный», 2014.- 36 с.
4. *Слепцова Л.Д.* Программирование на VBA в Microsoft Office 2010. – М.: Вильямс, 2010.
5. Информатика: Учебник для вузов / Под ред. Н.В. Макаровой – 3-е изд., перераб. - М.: Финансы и статистика, 2009. - 768 с.

Содержание