Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Санкт-Петербургский горный университет

Кафедра информатики и компьютерных технологий

ВВЕДЕНИЕ В ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ.

РАБОТА С ПОЛЬЗОВАТЕЛЬСКИМИ ФУНКЦИЯМИ, ПОДПРОГРАММАМИ И ДИАГРАММАМИ В СРЕДЕ VBA

Методические указания к практическим занятиям для студентов специальностей 21.05.02, 21.05.03

САНКТ-ПЕТЕРБУРГ
2022

УДК 004.432.42

ВВЕДЕНИЕ В ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. Работа с пользовательскими функциями, подпрограммами и диаграммами в среде VBA: Методические указания к практическим занятиям / Санкт-петербургский горный университет. Сост.: *С.Ю. Кротова, Е.Н. Овчинникова* СПб. 2022. 43 с.

В методическом пособии представлен теоретический материал и варианты индивидуальных заданий для выполнения студентами на практических занятиях. Представлены примеры решения задач по созданию пользовательских функций, вычислению матричных выражений с помощью подпрограмм и построению диаграмм средствами VBA.

Методические указания предназначены для студентов специальностей 21.05.02 «Прикладная геология», 21.05.03, «Технология геологический разведки ».

Табл. 4 Ил. 42 Библиогр.: 5 назв.

Научный редактор доц. *А.Е. Ильин*

Рецензент: *К. В. Столяров*

Введение

VBA (Visual Basic for Applications) — это язык программирования, встроенный во множество отдельных программ и прикладных пакетов — от приложений Microsoft Office (включая Microsoft Project и Microsoft Visio) и до таких мощных пакетов, как AutoCAD, CorelDraw и Adobe Creative Suite, не говоря уже о многочисленных специализированных приложениях, предназначенных для управления производственными процессами, учета финансовых ресурсов или информационной поддержки клиентов.

Помимо программирования линейных, разветвляющихся и циклических процессов, среда VBA позволяет создавать пользовательские функции и процедуры с подпрограммами и функциями для решения определённого класса задач. А так же решать задачи по визуализации данных при построении графиков и диаграмм.

В данном методическом пособии описаны методы создания пользовательских функций. Особенности работы с подпрограммами на примере выполнения операций с матрицами, а так же принцип построения диаграммы для функции, табулированной в заданном интервале с последующим запуском полученного программного кода с помощью пользовательской формы. А так же описаны способы построения диаграмм с помощью макроса, записанного средствами макрорекордера.

В каждом разделе приводятся примеры решения задач и предлагаются варианты для индивидуальной работы студентов.

1. Создание пользовательских функций

Функция **–** это подпрограмма, которая возвращает значение. Функции бывают:

- *встроенные (*стандартные) - являются частью языка и могут вызываться по имени без предварительного описания (abs(x), sqr(x));
- *функции пользователя* пишутся самим программистом. Их вызов для последующего выполнения записывается обычно в разделе операторов.

Структура функции

Function *имя [(список аргументов)]* [As *тип*]
операторы
[*имя = выражение*]
End Function

где:

имя — обязательный элемент. Содержит имя подпрограммы-функции Function, удовлетворяющее стандартным правилам именования переменных;

список аргументов — необязательный элемент, это список переменных, представляющий параметры, которые передаются в подпрограмму Function при ее вызове (формальные параметры). Имена переменных разделяются запятой.

тип — необязательный элемент. Тип данных значения, возвращаемого подпрограммой Function.

операторы— элемент, содержащий любую группу операторов, выполняемых внутри процедуры Function.

выражение — возвращаемое значение подпрограммой Function.

Пример 1. Выполнить табулирование функции

$$y = x + \frac{4}{x + 0{,}5}$$

в интервале [0, 10]**,** аргумент x меняется с шагом 0,5.

1. Открываем окно проекта VBA и вставляем функцию (*Insert-Procedure*), выбирая в открывшемся окне *Function* (Рис.1.1).

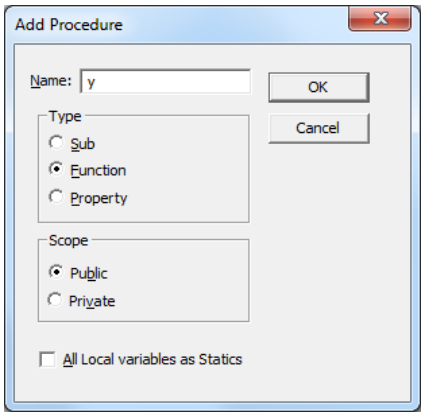


Рисунок 1.1. Создание функции

2. В открывшемся модуле записываем формулу задающую функцию y (рис 1.2)

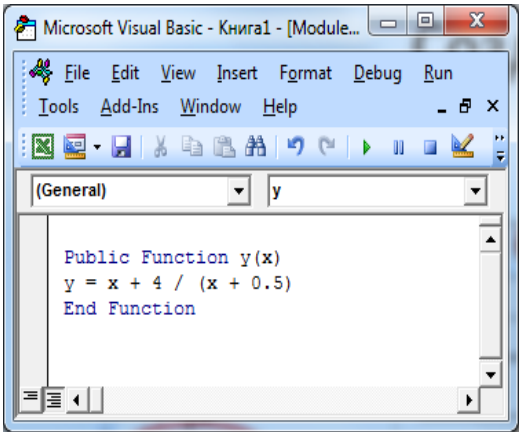


Рисунок 1.2. Создание функции

3. Вносим в ЭТ значения аргумента x, и вставляем созданную ранее функцию (*вставка-функция*) находим функцию y в разделе *определённые пользователем (рис. 1.3)*

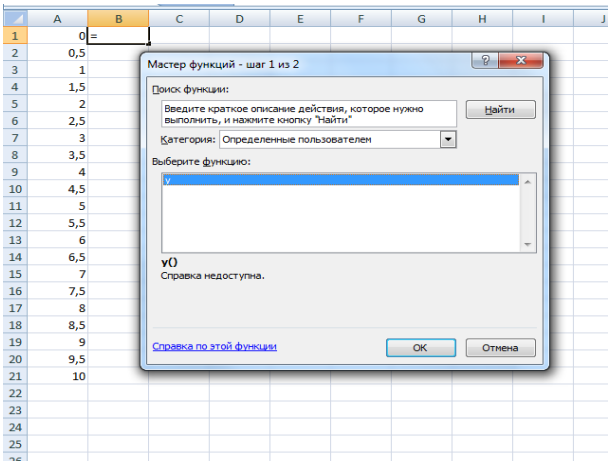


Рисунок 1.3. Вставка функции

4. В открывшемся окне вносим ссылку на ячейку с аргументом *x*(рис.1.4).

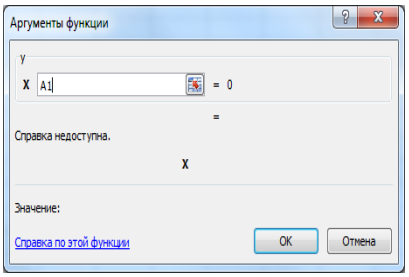


Рисунок 1.4. Окно пользовательской функции

5. Копируем вставленную функцию на заданный диапазон.

Пример 2. Выполнить табулирование функции

$$y(x) = \begin{cases} 2\cos(\pi x), & \cos(\pi x) \geq 0 \\ -\cos(\pi x), & \cos(\pi x) < 0 \end{cases}$$

Интервал изменения аргумента от 0 до 3, шаг 0,2.

Задача решается аналогично примеру 1, с той лишь разницей, что при создании пользовательская функция y задаётся от двух аргументов x и pi, поскольку в среде VBA не поддерживаются встроенные математические константы? F nfr ;t используются операторы условного перехода . На рисунке 1.5 показаны три варианта задания пользовательской функции для решения данной задачи.
При вставке созданной функции в ячейку ЭТ необходимо сделать относительную ссылку на аргумент x и абсолютную на аргумент Pi (рис.1.6)

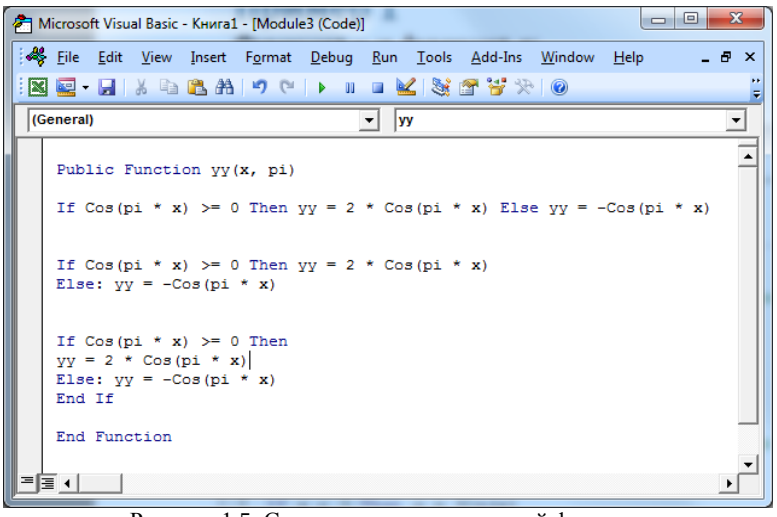


Рисунок 1.5. Создание пользовательской функции

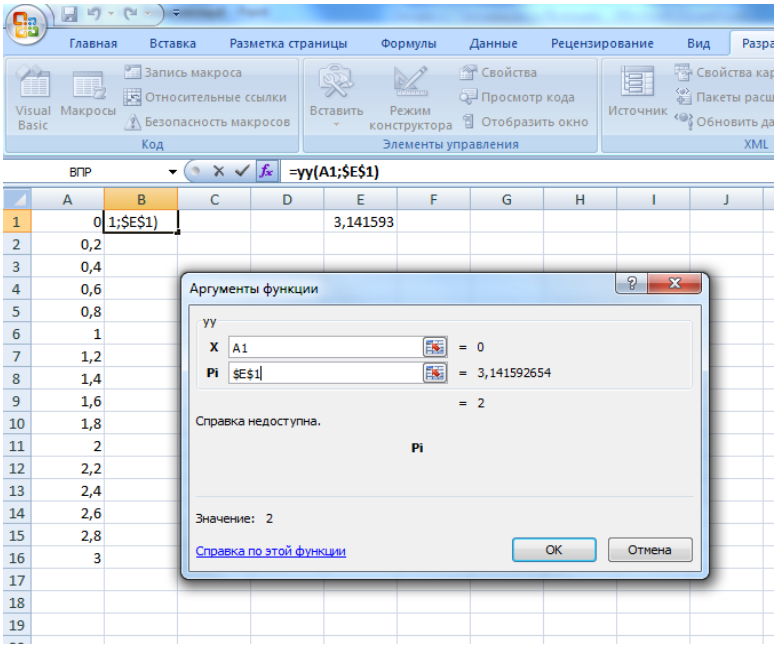


Рисунок 1.6. Окно пользовательской функции

Индивидуальные задания

Задание 1. Выполнить табулирование функции ***y*** при изменении аргумента ***x*** в заданном интервале с указанным шагом. Функцию задать двумя способами в виде формулы и с помощью создания пользовательской функции. Результаты сравнить. Варианты заданий приведены в таблице 1.

Таблица 1

Вариант	Функция	Интервал изменения аргумента	Шаг изменения аргумента
1	$y = x + \frac{4}{x + 0{,}5}$	[0, 10]	0,5
2	$y = 3 \cdot (x - \sin 2x)$	[-1,4]	0,25
3	$y = (x + 2) \cdot \sin 3x$	[-2,2]	0,2
4	$y = \frac{x - \sin 2x}{\lvert x \rvert + 1}$	[-4,4]	0,5
5	$y = (x + 0{,}5) \sin 2x$	[-2,2]	0,2
6	$y = (x - 1) \cdot e^{-x}$	[0,5]	0,25
7	$y = \sqrt{x} \cdot e^{-x}$	[0,4]	0,2
8	$y = (x^2 - x) \cdot e^{x}$	[-4,2]	0,25
9	$y = \frac{1 - \ln x}{1 + \ln x}$	[1,10]	0,5
10	$y = \frac{0{,}5 \cdot x^2 - x + 2}{x^2 + 1}$	[-10,10]	1
11	$y = \frac{x + 1}{x^2 + 1} e^{-x}$	[-2,3]	0,2
12	$y = \cos^2 2x - 3 \cdot \sin x$	[-3,3]	0,25
13	$y = 2^{-\lvert x \rvert} x$	[-3,3]	0,25
14	$y = \sqrt[3]{x} \sin x$	[-10,10]	1
15	$y = \sqrt[3]{x} \cos x$	[-5,5]	0,5

Задание 2. Выполнить табулирование функции ***y*** при изменении аргумента ***x*** в заданном интервале шаг задать самостоятельно. Функцию задать двумя способами в виде формулы и с помощью создания пользовательской функции. Результаты сравнить. Варианты заданий приведены в таблице 2.

Таблица 2

№	Функция	Интервал
1	$y(x)=\begin{cases}2\cos(\pi x), & \cos(\pi x)\geq 0\\ -\cos(\pi x), & \cos(\pi x)<0\end{cases}$	$x\in[0;3]$
2	$y(x)=\begin{cases}xsin(\pi x) & \sin(\pi x)\leq 0{,}5\\ 0{,}5, & \sin(\pi x)>0{,}5\end{cases}$	$x\in[0;3]$
3	$y(x)=\begin{cases}\sqrt{1-e^{x}}, & x\leq 0\\ \sqrt{x}, & x>0\end{cases}$	$x\in[-1{,}5;3]$
4	$y(x)=\begin{cases}cos^{2}(x), & \cos(x)\geq \ln(x)\\ ln^{3}(x), & \cos(x)<\ln(x)\end{cases}$	$x\in[0{,}1;3]$
5	$y(x)=\begin{cases}xsin^{2}(x) & \sin(x)<0\\ 0{,}5x, & \sin(x)\geq 0\end{cases}$	$x\in[-2;1{,}5]$
6	$y(x)=\begin{cases}\frac{1}{x} & \frac{1}{x}>\ln(x)\\ 10\ln(x)-5, & \frac{1}{x}<\ln(x)\end{cases}$	$x\in[0{,}1;3]$
7	$y(x)=\begin{cases}\sqrt{\lvert x\rvert}, & x<0\\ x^{2}, & x\geq 0\end{cases}$	$x\in[-2;1]$
8	$y(x)=\begin{cases}xe^{-x^{2}}, & xe^{-x}<0{,}1\\ 0{,}1e^{-x}, & xe^{-x}\geq 0{,}1\end{cases}$	$x\in[-1{,}5;1{,}5]$
9	$y(x)=\begin{cases}5e^{-x}\sin(\pi x), & x\geq 0\\ 0, & x<0\end{cases}$	$x\in[-0{,}5;2{,}5]$
10	$y(x)=\begin{cases}4x^{2}-3, & x\leq 1\\ \frac{1}{x^{2}}, & x>1\end{cases}$	$x\in[-0{,}5;2{,}5]$
11	$y(x)=\begin{cases}3sin(\pi x)-1{,}5 & \lvert\sin(\pi x)\rvert\geq 0{,}5\\ 0, & \lvert\sin(\pi x)\rvert<0{,}5\end{cases}$	$x\in[0;3]$
12	$y(x)=\begin{cases}x^{3}, & \lvert x\rvert\leq 1\\ x, & \lvert x\rvert>1\end{cases}$	$x\in[-1{,}5;1{,}5]$
13	$y(x)=\begin{cases}3x^{4}+2, & x\leq 1\\ \frac{1}{x^{2}}, & x>1\end{cases}$	$x\in[-0{,}5;2{,}5]$
14	$y(x)=\begin{cases}\cos(\pi x), & \cos(\pi x)\geq 0\\ -\cos(\pi x), & \cos(\pi x)<0\end{cases}$	$x\in[0;4]$

№	Функция	Интервал
15	$(x) = \begin{cases} 2xsin^2(x) & \sin(x) < 0 \\ 0{,}7x\,, & \sin(x) \geq 0 \end{cases}$	$x \in [0; 4]$
16	$y(x) = \begin{cases} \sqrt{\lvert x \rvert}, & x < 0 \\ x^2 \quad , & x \geq 0 \end{cases}$	$x \in [-0{,}5; 2{,}5]$

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.

Процедура имеет следующую структуру:

Sub *имя [(список аргументов)]* [As *тип*]
операторы
End Sub

где:

имя — имя процедуры Sub, удовлетворяющее стандартным правилам именования переменных;

список аргументов — необязательный элемент, список переменных, представляющий параметры, которые передаются в процедуру Sub при ее вызове (формальные параметры), имена переменных разделяются запятой;

тип — необязательный элемент, тип формальных параметров;

операторы— любая группа операторов, выполняемых в процедуре Sub, находятся между Sub и End Sub.

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)

На рисунке 2.1 представлена процедура для реализации теоремы Пифагора, являющаяся подпрограммой для вычисления и вывода данных. Здесь в скобках после имени процедуры указываются *формальные* параметры, которые при обращении к этой процедуре основной подпрограммой будут заменены на *фактические.*

```
Sub pifagor(x, y, z)
z = Sqr(x * x + y * y)
Cells(1, 3).Value = z
End Sub
```

Рисунок 2.1. Пример процедуры (подпрограмма)

Основная процедура представлена на рисунке 2,2. Здесь при обращении к подпрограмме в скобках указываются *фактические параметры.*

```
Sub Main()
a = Range("A1")
b = Range("B1")
Call pifagor(a, b, c)
End Sub
```

Рисунок 2.2. Основная процедура

Пример 3. Создать процедуру, вычисляющую площадь круга по значению радиуса R (Рис. 2.3).

```
Public Sub Обращение()
Dim Радиус As Variant

' Задание первого значения радиуса
Радиус = 25
    ' Обращение к программе площади круга
    ' вычисление для радиуса =25
       Call ПлощадьКруга(Радиус)

    ' вычисление для радиуса =18
    Call ПлощадьКруга(18)
End Sub
```

```
    'вспомогательная программа
    Sub ПлощадьКруга(R)
    Dim S  As Variant
    S = 3.14 * R ^ 2
    MsgBox ("Для R = " & R & " Площадь круга равна " & S)
    End Sub
```

Рисунок 2.3. Пример программы

Пример 4. Вычислить сумму элементов двух матриц A(3,3) и B (3,3). Для ввода, вывода и суммирования матриц использовать отдельные процедуры.

Для решения данной задачи необходимо составить подпрограммы для выполнения отдельных операций (ввод, суммирование, вывод) и основную процедуру, связывающую эти вспомогательные подпрограммы.

Процедура для ввода матриц изображена на рисунке 2.4. 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 2.4. Процедура для ввода матриц

Процедура для суммирования изображена на рисунке 2.5.

```
  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 2.5. Процедура для сумирования матриц

Процедура вывода результирующей матрицы представлена на рисунке 2.6. Здесь переменная ***nam*** это текстовый комментарий, который вставляется в соответствующую ячейку.

Основная процедура представлена на рисунке 2.7. В скобках указаны фактические параметры, которые принимают значения формальных параметров-соответствующих переменных, объявленных в описании данной процедуры.

```
    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 2.6. Процедура для сумирования матриц

```
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 2.7 Основная прроцедура

Пример 5. Выполнить транспонирование матрицы A(3,3). На рисунке 2.8 и 2.9 представлен листинг программы для транспонирования заданной матрицы. В подпрограмме для

транспонирования строка ***M2(i, j) = M1(j, i)*** непосредственно выполняет замену строк и столбцов 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 2.8 Подпрграммы для ввода, выводла и транспонирования 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 2.9 Основная процедура для транспонирования матрицы

Пример 6. Выполнить матричное умножение матриц A(4,3) и B(3,2). На рисунке 2.10 и 2.11 представлен листинг программы для решения задачи. Здесь в подпрограмме для выполнения матричного умножения необходимо ввести дополнительный цикл по переменной r которая меняется от 1 до общего числа строк и столбцов перемножаемых матриц.

```
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
```

Рисунок 2.10. Основная процедура

```
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

  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 2.11. Подпрограммы для ввода, умножения и вывода

Пример 7. Умножить матрицу A(3,3) на число 5.
Листинг программы представлен на рисунках 2.12 и 2.13.

```
    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
    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
```

Рисунок 2.12. Подпрограммы для ввода, умножения и вывода

```
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 2.13 Основная процедура

Индивидуальные задания

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

Таблица 3

№	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

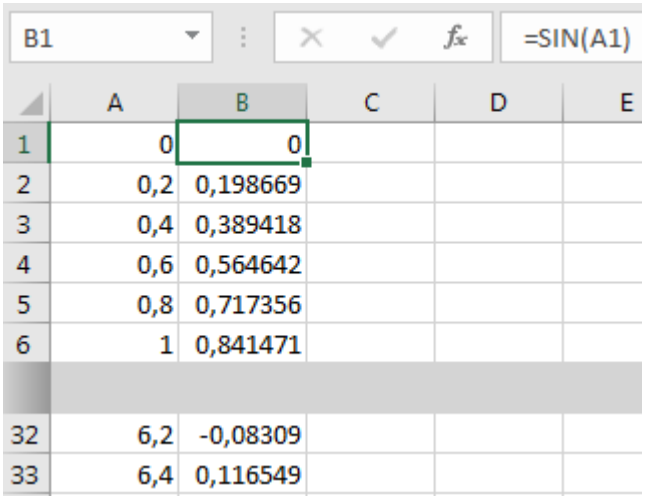
Чтобы построить график средствами VBA, необходимо:

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. Табулирование значения аргумента с заданными параметрами в диапазоне ячеек A1:A33 и расчет по ним значения функции по адресам B1:B33 (рис. 3.1).



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. Построение простейшей диаграммы на текущем листе MS Excel (***ActiveSheet***). Для этого 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. Определение источника 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 рисунке 3.3. Поведение функции на нем уже заметно, но, прежде чем использовать диаграмму на экранной форме или текстовом документе, необходимо выполнить её форматирование.

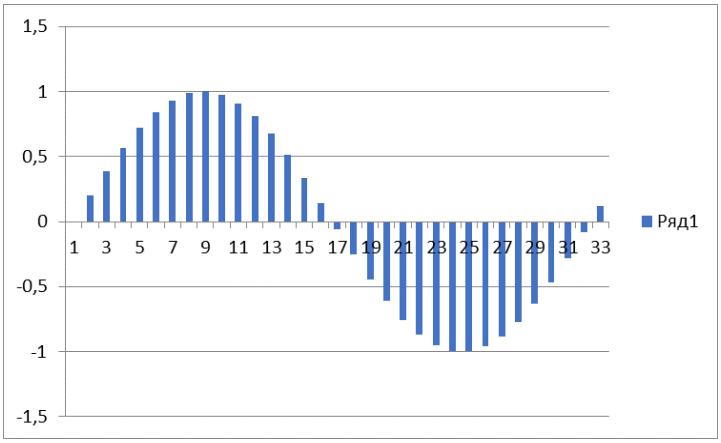


Рис.3.3 Простейший график функции

5. Для текущей (активной) диаграммы ***ActiveChart,*** для ее свойства ***ChartType*** выбрать нужный тип диаграммы из предусмотренных в *VBA* (рис. 3.4).

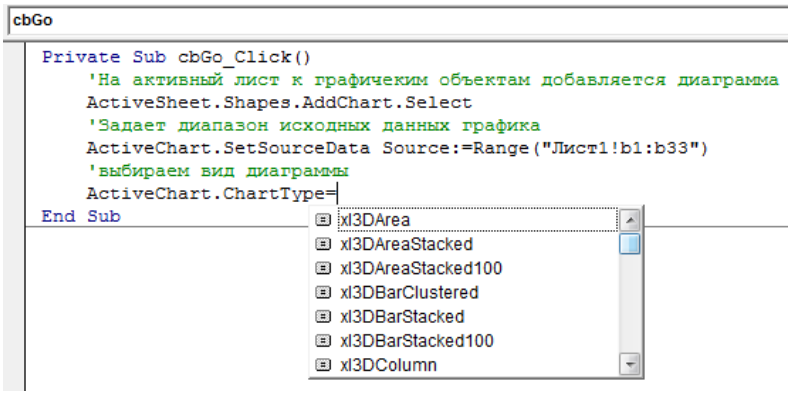


Рисунок 3.4 Изменение типа диаграммы

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

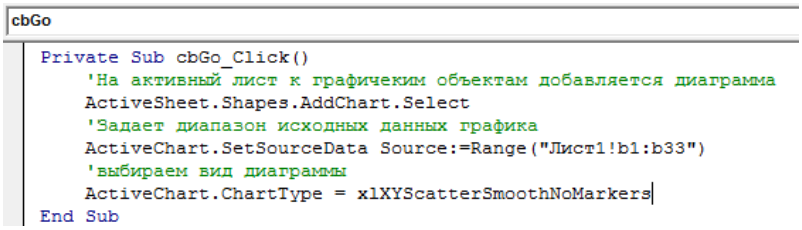


Рисунок 3.. 5 Изменение типа диаграммы на xlXYScatterSmoothNoMarkers

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

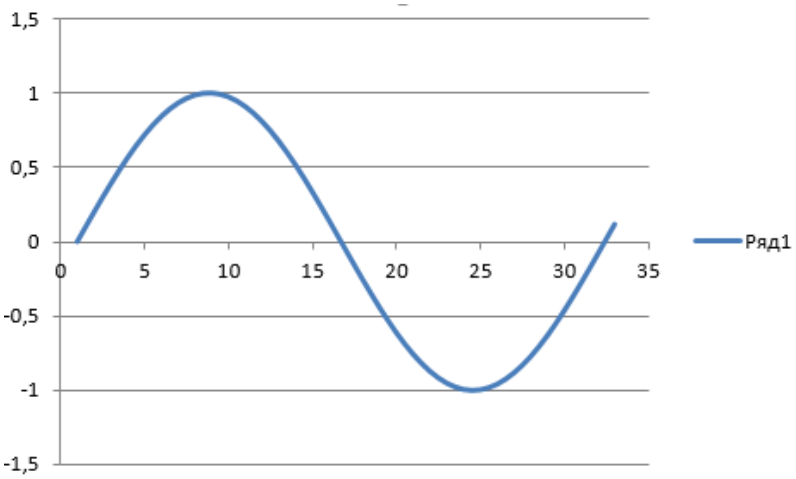


Рисунок 3..6 Вид диаграммы со значением ActiveChart.ChartType = xlXYScatterSmoothNoMarkers

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

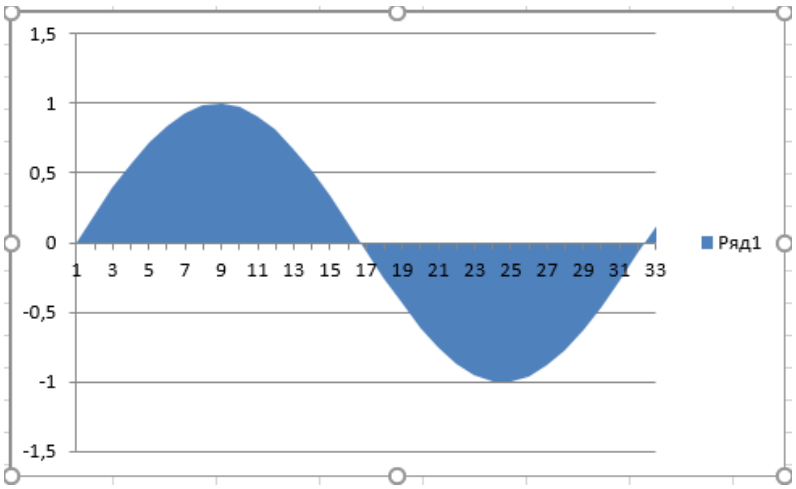


Рисунок 3..7 Вид диаграммы со значением ActiveChart.ChartType = xlArea

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).

```
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")
    ActiveChart.FullSeriesCollection(1).XValues = ("Лист1!a1:a33")
    'выбираем вид диаграммы
    ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
    'отображать названия осей графика
    ActiveChart.Axes(xlValue).HasTitle = True
    ActiveChart.Axes(xlCategory).HasTitle = True
End Sub
```

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.

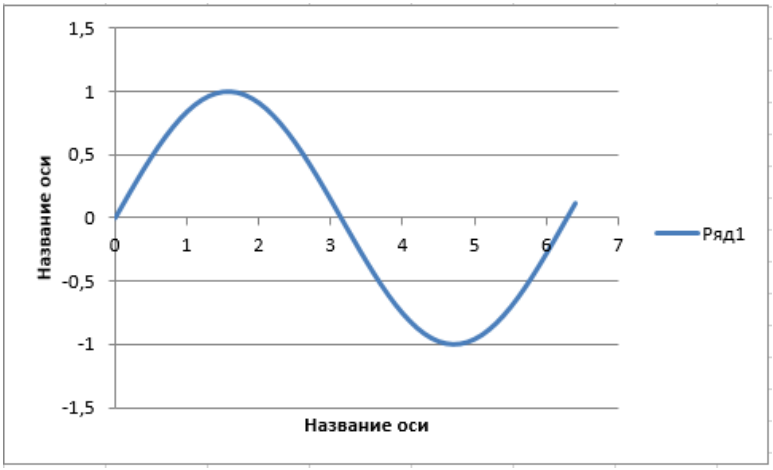


Рисунок .3. 11 Вид диаграммы со значениями по оси x и видимыми названиями осей

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.

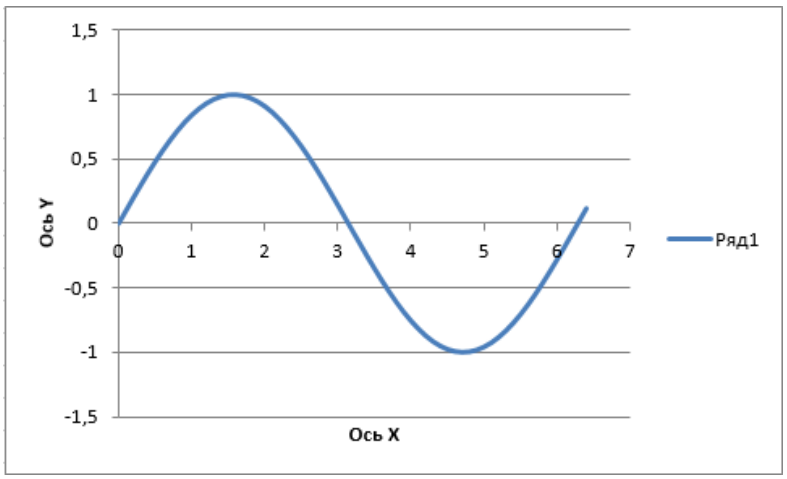


Рисунок 3. .13 Вид диаграммы с подписями по осям

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.

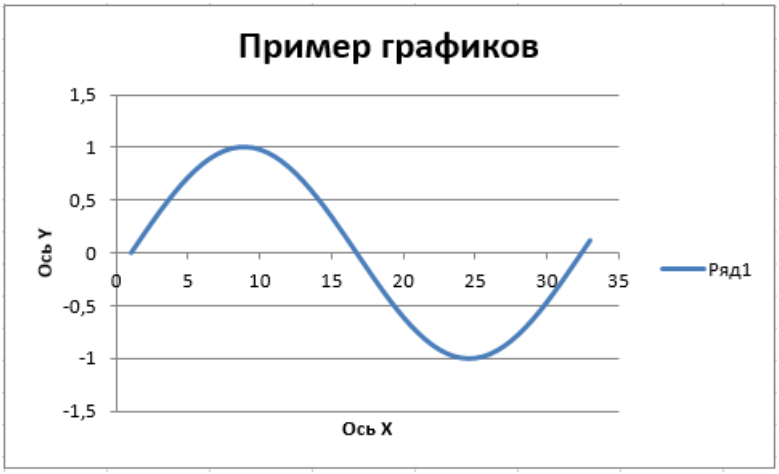


Рисунок 3..15 Вид диаграммы с измененным заголовком

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.

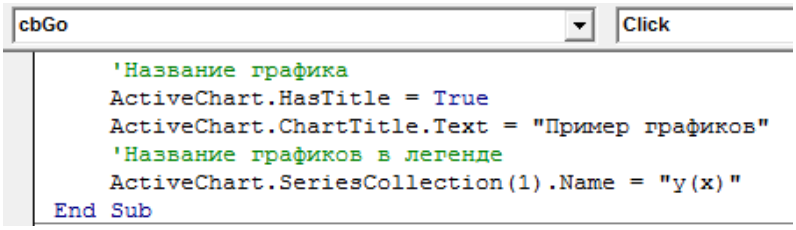


Рисунок 3.16 Изменение легенды диаграммы

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.

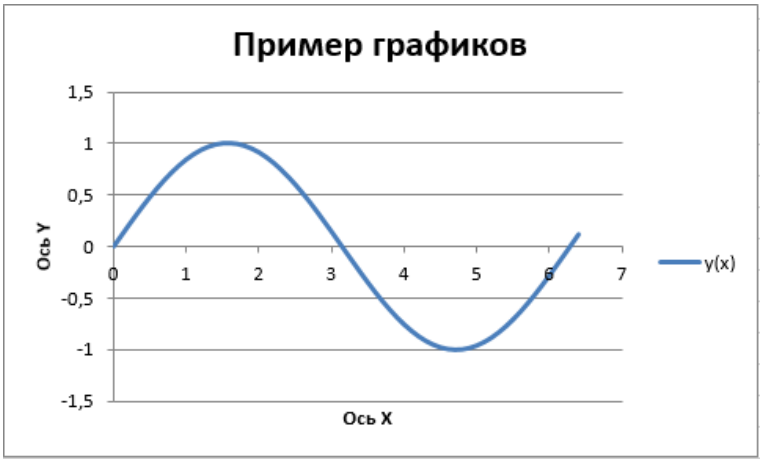


Рисунок 3.17 Окончательный вид диаграммы, построенной средствами VBA

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")

Часто требуется указать путь файла полностью:

imgGraph.Picture = LoadPicture("C:\Users\Дом \Мои документы\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:

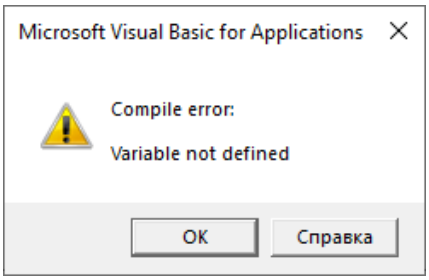


Рис 3. 18 Сообщение об ошибке при неправильном имени объекта

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.

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.

Для наиболее полного решения задачи, в программный код, работающий при нажатии командной кнопки «Построить» необходимо добавить фрагмент табулирования функции в указанном интервале. Полученный программный код представлен на рисунке 3.19.

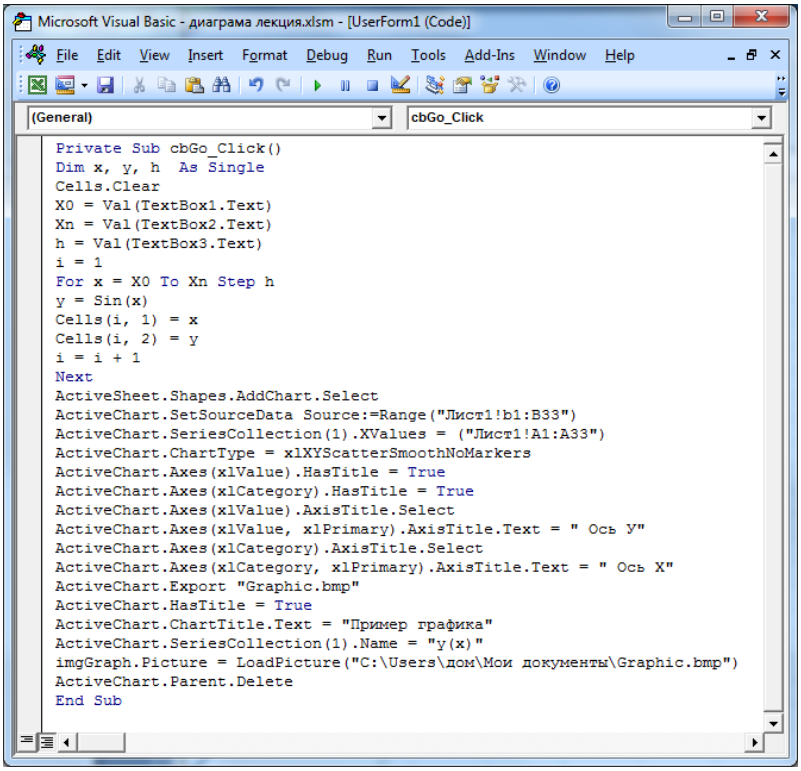


Рисунок 3.19 Программный код

Конечный результат работы программы представлен на рисунке 3.20:

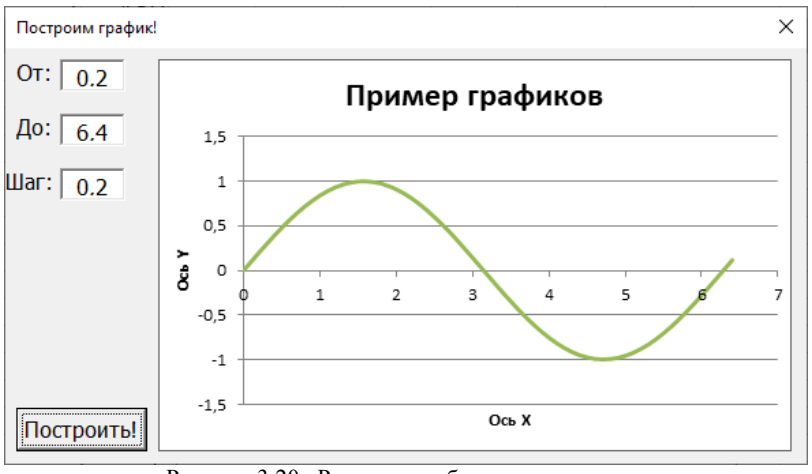


Рисунок 3.20. Результат работы программы

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).

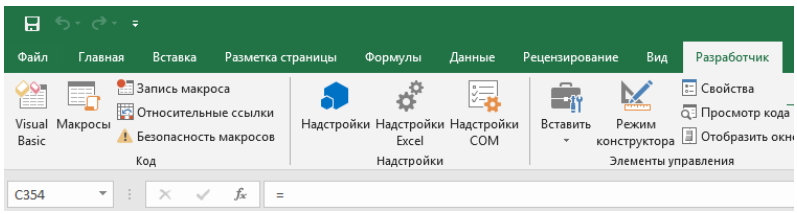


Рисунок 3. .20 Место кнопки записи макроса

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).

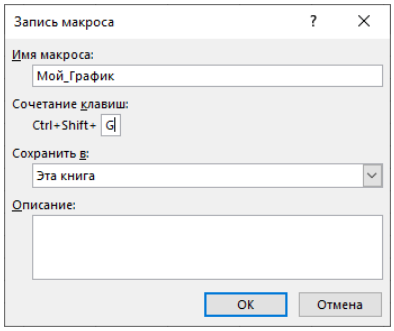


Рисунок 3..21 Окно свойств записи макроса

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):

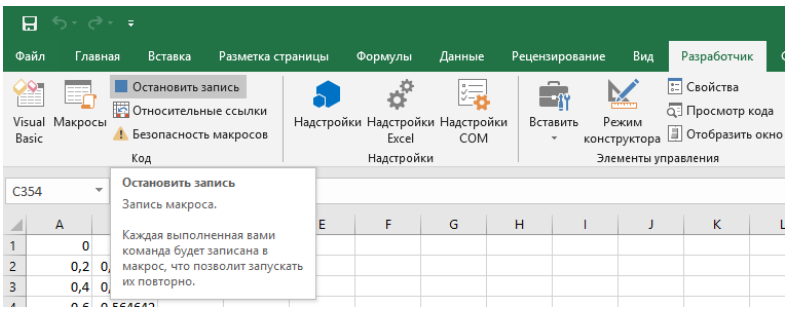


Рисунок 3. .22 Место кнопки остановки записи макроса

В результате получается подпрограмма на VBA, записанная в отдельный модуль. Ее можно посмотреть и отредактировать, открыв среду разработки(Рис 3.23).

```
Sub Макрос5()
'
' Макрос5 Макрос
'
'
    ActiveCell.FormulaR1C1 = "0"
    Range("A2").Select
    ActiveCell.FormulaR1C1 = "0.2"
    Range("A1:A2").Select
    Selection.AutoFill Destination:=Range("A1:A33"), Type:=xlFillDefault
    Range("A1:A33").Select
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "=SIN(RC[-1])"
    Range("B1").Select
    Selection.AutoFill Destination:=Range("B1:B33"), Type:=xlFillDefault
    Range("B1:B33").Select
    Range("A1:B33").Select
    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.SetSourceData Source:=Range("'Лист1'!$A$1:$B$33")
    ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
    ActiveChart.SetElement (msoElementChartTitleCenteredOverlay)
    ActiveSheet.ChartObjects("Диаграмма 1").Activate
    ActiveChart.SetElement (msoElementPrimaryCategoryAxisTitleAdjacentToAxis)
    ActiveSheet.ChartObjects("Диаграмма 1").Activate
    ActiveChart.SetElement (msoElementPrimaryValueAxisTitleHorizontal)
    ActiveSheet.ChartObjects("Диаграмма 1").Activate
    ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Text = "y(x)"
    ActiveSheet.ChartObjects("Диаграмма 1").Activate
    ActiveChart.ChartArea.Select
    ActiveSheet.ChartObjects("Диаграмма 1").Activate
    ActiveChart.SeriesCollection(1).Name = "=""y(x)"""
End Sub
```

Рисунок 3..23. Программный код макроса

Для запуска макроса необходимо во вкладке ***Разработчик*** выбрать ***Макрос.*** В открывшемся окне

выбрать имя нужного макроса и нажать на кнопку выполнить(рис. 3.24)

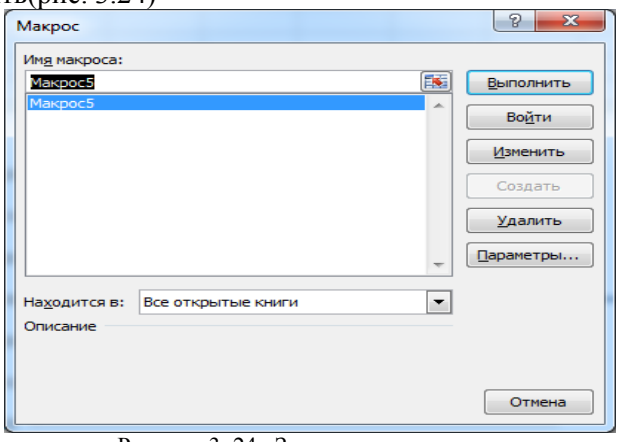


Рисунок 3..24. Запуск макроса

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.

Индивидуальные задания

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.

- Построить диаграмму заданной функции средствами макрорекордера и сравнить полученные результаты.

Таблица 4

№	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
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
№	Уравнение	**a**	**b**	h

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

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

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. *Слепцова Л.Д.* Программирование на VBA в Microsoft Office 2010. – М.: Вильямс, 2010.
4. Информатика: Учебник для вузов / Под ред. Н.В. Макаровой – 3-е изд., перераб. - М.: Финансы и статистика, 2009. - 768 с.
5. *Ильин А.Е., Кротова С.Ю., Чиргин А.В.* Информатика. Программирование на VBA. Обработка и визуализация данных. Методические указания к лабораторным работам для студентов специальностей «Стандартизация и метрология», «Электроэнергетика и электротехника», «Техносферная безопасность».. СПб: Санкт-петербургский горный университет. 2019.-45 с.

Содержание