Программирование в VBA

Оглавление

Понятие макроса.

Для автоматизации работы в текстовом редакторе или в табличном процессоре часто применяются макросы. Необходимость использования макросов возникает при многократном повторении нескольких команд или для программирования действий, не заложенных в программе. Вместо того чтобы вручную делать отнимающие много времени и повторяющиеся действия, можно создать и запускать один макрос, который будет выполнять эту задачу.

Назначение макросов

Макросы часто используются для следующих целей:

- для ускорения часто выполняемых операций редактирования или форматирования;
- для объединения нескольких команд, например, для вставки таблицы с указанными размерами и границами и определенным числом строк и столбцов;
- для упрощения доступа к параметрам в диалоговых окнах;
- для автоматизации обработки сложных последовательных действий в задачах.

Для создания макроса в Word можно использовать два метода: с помощью средства для записи макросов (макрорекордера) или редактора Visual Basic.

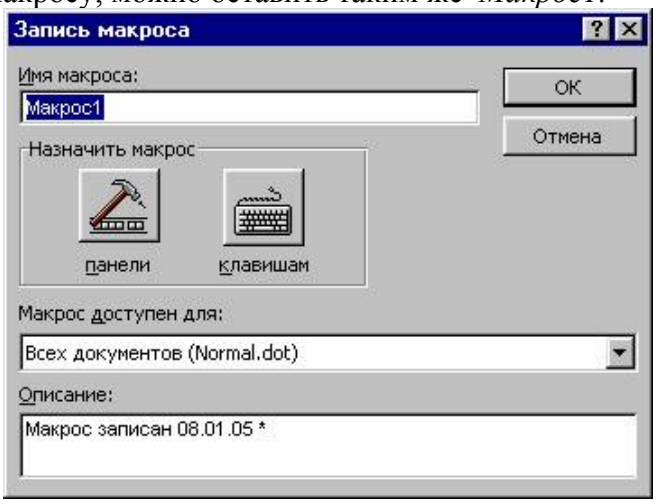
Запись макроса с помощью макрорекордера

Средство записи макросов позволяет быстро создавать макросы с минимальными усилиями. При этом макрос записывается в виде последовательности инструкций на языке программирования Visual Basic для приложений. Допускается применение мыши для выбора команд и параметров. Однако действия, проделанные в окне документа с помощью мыши, не записываются. Например, с помощью мыши нельзя перемещать курсор, копировать и перемещать объекты, в том числе перетаскиванием. Для записи этих действий используйте клавиатуру. Запись макроса можно временно приостанавливать и затем возобновлять с того места, где запись была остановлена.

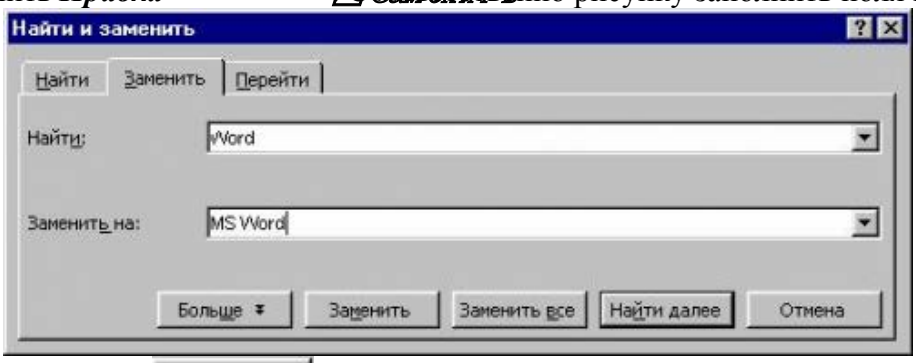
Пример

Пример. Разработать макрос, заменяющий слово Word на MS Word.

1. Выполнить ***Сервис → Макрос → Начать запись***
2. Присвоить имя макросу, можно оставить таким же *Макрос1*.



3. Нажать ОК.
4. На экране появляется панель  и измениться курсор.
5. Выполнить ***Правка → Заменить***. Согласно рисунку заполнить поля в окне.



6. Нажать на кнопку  .
7. Закрыть окно *Найти и заменить*.
8. Остановить запись через меню ***Сервис → Макрос → Остановить запись*** или через панель  , нажав на кнопку  .

Для быстрого запуска макроса можно назначить ему кнопку панели инструментов, команду меню или сочетание клавиш. После этого для выполнения макроса будет

достаточно выбрать команду в меню, нажать кнопку на панели инструментов или нажать сочетание клавиш. Чтобы запустить макрос, можно также выбрать команду ***Макрос*** в меню***Сервис***, команду ***Макросы***, а затем — имя макроса, который требуется выполнить.

Хранение макросов

Макросы хранятся в шаблонах или в документах. По умолчанию макросы сохраняются в шаблоне *Normal.dot*, чтобы они были доступны всем документам Word. Если макрос будет использоваться только в одном документе, то его необходимо сохранить именно в этом документе. Отдельные макросы в документе хранятся в проектах макросов, которые можно копировать из одного документа в другой. Для копирования, удаления или переименования проекта макросов используют организатор (***Сервис → Макрос → Макросы***, кнопка ***Организатор***).

Выполнение (Удаление) макроса

1. Выбрать ***Сервис → Макрос → Макросы***
2. Выбрать имя макроса, который требуется выполнить, в списке *Имя*.
3. Если нужного макроса нет в списке, выбрать другой документ, шаблон или список в списке *Макросы из*.
4. Нажать кнопку ***Выполнить (Удалить)***.

Просмотр (редактирование) программного кода макроса

1. Выбрать ***Сервис → Макрос → Макросы***
2. Выбрать имя макроса, который требуется просмотреть, в списке *Имя*.
3. Если нужного макроса нет в списке, выбрать другой документ, шаблон или список в списке *Макросы из*.
4. Нажать кнопку ***Изменить***.

Организация ввода и вывода сообщений

Подобно многим языкам программирования Visual Basic for Application (VBA) позволяет создать три типа процедур: Sub, Function, Property.

Процедура – это набор описаний и инструкций, сгруппированных для выполнения.

Процедура Sub – набор команд, с помощью которого можно решить определенную задачу. При ее запуске выполняются команды процедуры, а затем управление передается в приложение или процедуру, которая вызвала процедуру Sub. Записываемые макросы автоматически описываются как процедуры Sub, любой макрос или другой код VBA, который просто выполняет определенный набор действий, используя приложенияOffice, и обычно является процедурой Sub.

Процедура Function (или функция) также представляет собой набор команд, который решает определенную задачу. Различия заключается в том, что процедуры данного типа обязательно возвращают значение. При создании процедуры Function можно описать тип данных, который возвращает функция. Функции обычно используются при выполнении вычислений, операциями с текстом, либо возвращают логические значения.

Процедура Property используется для ссылки на свойство объекта. Данный тип процедур применяется для установки или получения значения пользовательских свойств форм и модулей. Процедуры облегчают хранение и применение информации, если использовать их сначала для сохранения в свойстве этой информации, а затем для ее чтения.

Структура процедуры

При записи процедуры требуется соблюдать правила ее описания. Упрощенный синтаксис для процедур Sub является следующим:

Sub имя ([аргументы])
Инструкции
End Sub

Синтаксис описания функций очень похож на синтаксис описания процедуры Sub, однако, имеются некоторые отличия:

Function *имя* ([*аргументы*]) [**As** *Тип*]
Инструкции
имя = выражение
End Function

Использование операторов

Процедуры состоят из операторов – наименьших единиц программного кода. Как правило, операторы занимают по одной строке программного кода, и в каждой строке обычно содержится только один оператор, но это не обязательно. В VBA имеется четыре типа операторов: объявления, операторы присваивания, выполняемые операторы и параметры компилятора.

Объявления

Объявление – это оператор, сообщающий компилятору VBA о намерениях по поводу использования в программе именованного объекта (переменной, константы, пользовательского типа данных или процедуры). Кроме того, объявление задает тип объекта и обеспечивает компилятору дополнительную информацию о том, как использовать данный объект. Объявив объект, можно использовать его в любом месте программы.

Переменные – это именованные значения, которые могут изменяться во время выполнения программы.

Рассмотрим пример объявления переменной.

С помощью оператора Dim объявляется переменная с именем *МоеЛюбимоеЧисло* и объявляется, что значение, которое она будет содержать, должно быть целым:

Dim МоеЛюбимоеЧисло As Integer

Константы представляют собой именованные значения, которые не меняются.

Оператор Constant создает строковую константу (текст) с именем *НеизменныйТекст*, представляющую собой набор символов *Вечность*:

Constant НеизменныйТекст = "Вечность"

Оператором Type объявляется пользовательский тип данных с именем *Самоделкин*, определяя его как структуру, включающую строковую переменную с именем *Имя* и переменную типа *Date* с именем *ДеньРождения*. В данном случае объявление займет несколько строк:

Type Самоделкин
Имя As String
ДеньРождения As Date
End Type

Объявление Private создает процедуру типа Sub с именем *СкрытаяПроцедура*, говоря о том, что эта процедура является локальной в смысле области видимости. Завершающий процедуру оператор End Sub считается частью объявления.

Private Sub СкрытаяПроцедура ()
инструкции

End Sub

Оператор присваивания

Оператор присваивания = приписывают переменным или свойствам объектов конкретные значения. Такой оператор всегда состоят из трех частей: имени переменной, или свойства, знака равенства и выражения, задающего нужное значение.

Оператор = присваивает переменной *МоеЛюбимоеЧисло*значение суммы переменной *ДругоеЧисло* и числа *12*.

МоеЛюбимоеЧисло = ДругоеЧисло + 12

В следующей строке кода, записывается, что свойству*Color (Цвет)* объекта *AGraphicShape* присваивается значение*Blue (Синий)* в предположении, что *Blue* является именованной константой:

AGraphicShape.Color = Blue

В следующеей строке, чтобы задать значение переменной *КвадратныйКорень*, для текущего значения переменной *МоеЛюбимоеЧисло*вызывается функция *Sqr* — встроенная функция VBA вычисления квадратного корня:

КвадратныйКорень = Sqr (МоеЛюбимоеЧисло)

В VBA выражением называется любой фрагмент программного кода, задающий некоторое числовое значение, строку текста или объект. Выражение может содержать любую комбинацию чисел или символов, констант, переменных, свойств объектов, встроенных функций и процедур типа Function, связанных между собой знаками операции (например, + или *). Несколько примеров выражений:

Выражение	**Значение**
3.14	3.14
Xn*5	10 (в предположении, что Xn = 2)
(12 - Sqr(x))/5	2 (в предположении, что x = 4)
«Розы красные,» & « фиалки фиолетовые»	Розы красные, фиалки фиолетовые

Выполняемые операторы

Выполняемые операторы делают главную работу в программе и используются для выполнения следующих задач:

- вызов процедуры;
- активизация метода некоторого объекта;
- управление порядком, в котором должны выполняться другие операторы, посредством организации циклов или выбором участка программного кода (из нескольких альтернатив) для последующего выполнения;
- выполнение одного из встроенных операторов VBA или функции.

Пример. Оператор, вызывающий для выполнения метод Rotate объекта AGraphicShape:

AGraphicShape. Rotate(90)

Запись нескольких операторов

Как правило, каждый оператор занимает одну строку программного кода, но VBA не обязывает уместить оператор в одной строке. Если оператор слишком длинный, можно разместить его в двух или более строках, добавив в конце каждой из строк (кроме последней) символ подчеркивания (_).

Можно сделать и наоборот — разместить несколько операторов в одной строке программного кода. Например,

Dim A As Integer, B As Integer: A = 3: B = 5: A = A +B

Эта строка программного кода эквивалентна следующим четырем строкам:

```
Dim A As Integer, B As Integer
A = 3
B = 5
A = A + B
```

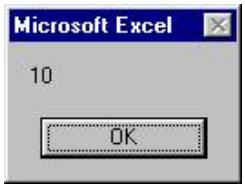
Самыми простыми диалоговыми окнами являются окна сообщений (message boxes) — это диалоговые окна, которые выдают пользователю сообщения и снабжаются одной или более кнопками для выбора. В VBA они создаются с использованием функции MsgBox.

В своей самой простой форме MsgBox используется как оператор с одним аргументом – сообщением, которое должно отображаться. Например, приведенный ниже макрос создаёт сообщение, показанное на рисунке.

```
Sub Program ()
MsgBox "Это - окно сообщений"
End Sub
```

MsgBox можно использовать для отображения числового значения.

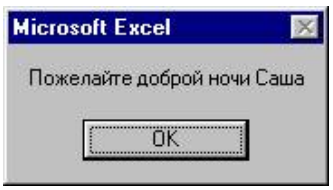
```
Sub ShoeValue()
Amount = 10
MsgBox Amount
End Sub
```



Переменной *Amount* присваивается значение 10. На следующей строке для отображения значения *Amount*используется MsgBox. Вокруг *Amount* нет кавычек, поскольку это – значение переменной, которое нужно выдать на экран, а не слово "Amount".

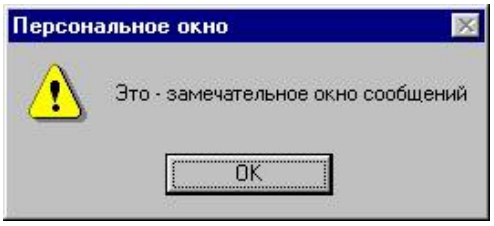
Чтобы использовать вместе две отдельные строки в одном окне сообщения, следует использовать операцию конкатенации (&) — объединение.

```
Sub SayGoodNight()
Name = "Саша"
MsgBox "Пожелайте доброй ночи " & Name
End Sub
```



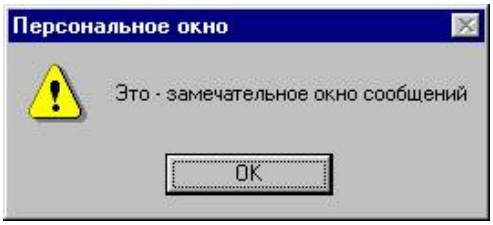
Переменной *Name* присваивается строка "Саша". В строке кода с MsgBox задаётся текстовая строка "Пожелайте доброй ночи ", за которой следует & *Name*, указывая MsgBox присоединить значение переменной *Name* к предыдущей текстовой строке.

Опции MsgBox



необязательные аргументы, например, для того, чтобы вставить значок или изменить заголовок (title).

MsgBox "Это - замечательное окно сообщений", _
vbExclamation, "Персональное окно"



Существует четыре значка для окон сообщений. Каждый имеет определённое числовое значение, которое должно передаваться в качестве аргумента MsgBox. Однако вместо числа можно использовать константы со специальными именами, встроенные в VBA.

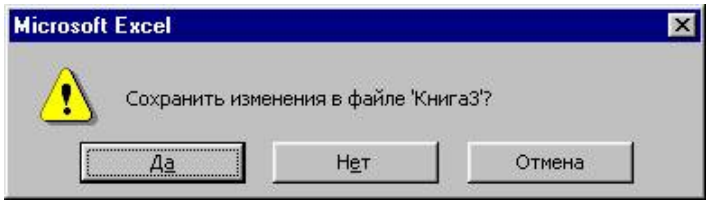
Таблица 1

Значки окна сообщений MsgBox

Отображение	Константа	Когда используется
	vbInformation	для сообщения, не требующего ответа
	vbQuestion	для того, чтобы задать вопрос
	vbExclamation	для выдачи важной информации
	vbCritical	для предупреждения

MsgBox как функция

MsgBox является функцией и может возвращать значение, соответствующее той кнопке, которую нажимает пользователь. Одной из имеющихся опций окна сообщения является изменение кнопок, которое оно отображает. Вот окно сообщений, которое появляется, когда осуществляется выход из Excel, имея не сохраненные изменения в документе. Это окно имеет три кнопки.



После выбора соответствующей кнопки Excel получает информацию о том, какую кнопку выбрали.

Общий формат для функции MsgBox:

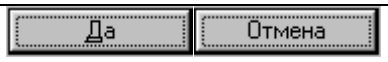
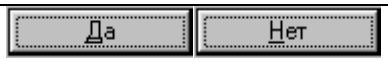
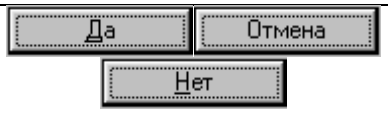
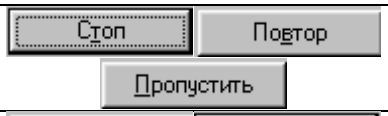
MsgBox(prompt [, buttons] [, title])

где *prompt* — единственный обязательный аргумент. Для окна сообщений следует задавать текстовую строку с информацией. если вы хотите изменить заголовок, появляющийся в верхней части окна, задайте для заголовка (*title*) текстовую строку. По умолчанию используется заголовок Microsoft Excel.

Таблица 2

Комбинации кнопок MsgBox

Отображение	Константа	Когда используется
	vbOKOnly	Когда не требуется от пользователя принятия решения

	vbOKCancel	Когда окно сообщений объясняет возможное действие. Позволяет пользователю сделать выбор с помощью кнопки *Отмена*
	vbYesNo	Альтернатива константе *vbOKCancel*, когда кажется, что это сделает окно сообщений более понятным
	vbYesNoCancel	Для таких ситуаций, как выход или закрытие файлов без сохранения (подобно ситуации, показанной на рисунке выше)
	vbAbortRetryIgnore	При ответе на сообщения об ошибках диска или файла
	vbRetryCancel	При ответе на сообщения об ошибках диска или файла

Если не указывать, какие кнопки необходимо отображать в окне сообщений, то используется значение по умолчанию, соответствующее кнопке ***Ok***.

Примеры решения задач

Приведем несколько примеров решения задач на VBA.

Пример 1. Вычислить значение выражения *a* равного $2 \cdot x - 3 \cdot y$, при *x* = 3, *y* = 2.5

Решение.

```
Sub выражение1()
Dim A, x, y
x = 3
y = 2.5
A = 2 * x - 3 * y
MsgBox (A)
End Sub
```

Пояснение решения.

В строке Dim A, x, y объявляются переменные *A*, *x*, *y*.

Пример 2. Вычислить значение выражения *a* равного $2 \cdot x - 3 \cdot y$, при *x* = 3, *y* = 2.5

Замечание: значения *x* и *y* вводит пользователь.

Решение.

```
Sub выражение2()
Dim A, x, y As Double
x = InputBox("Введите x=")
y = InputBox("Введите y=")
A = 2 * x - 3 * y
MsgBox (A)
End Sub
```

Пояснение решения.

В строке Dim A, x, y As Double описываются переменные *A*, *x*, *y* как числа двойной точности.

При использовании строки

```
x = InputBox("Введите x=")
```

появиться окно

Пример 3. Вычислить значение выражения *a* равного
$2 \cdot x - 3 \cdot y$, при $x = 3$, $y = 2.5$
Замечание: значения *x* и *y* вводит пользователь, ответ выводится в виде «a =<значение>».
Решение.

```
Sub выражение3()
Dim A, x, y As Double
Dim ответ As String
x = InputBox("Введите x=")
y = InputBox("Введите y=")
A = 2 * x - 3 * y
ответ = "a=" + Str(A)
MsgBox (ответ)
End Sub
```

Пояснение решения.
В строке Dim ответ As String описывается переменная *ответ* как строковая.
Код Str(A) преобразует значение переменной *A* в строку.

Пример 4. Вычислить значения выражений при $x = 3$, $y = 2.5$
$2 \cdot x - 3 \cdot y$,
$\frac{2 \cdot x - 3 \cdot y}{2}$,
$\frac{\frac{2 \cdot x - 3 \cdot y}{2}}{x}$,
$\frac{\frac{2 \cdot x - 3 \cdot y}{2}}{2 \cdot x} + \frac{5 - x}{3 + y}$
Решение.

```
Sub выражение4()
Dim A, b, c, d, a1, x, y As Double
x = InputBox("Введите x=")
y = InputBox("Введите y=")
A = 2 * x - 3 * y
b = (2 * x - 3 * y) / 2
c = (2 * x - 3 * y) / 2 * x
d = (2 * x - 3 * y) / (2 * x)
a1 = (2 * x - 3 * y) / (2 * x) + (5 - x) / (3 + y)
MsgBox ("a=" + Str(A))
MsgBox ("b=" + Str(b))
MsgBox ("c=" + Str(c))
MsgBox ("d=" + Str(d))
MsgBox ("a1=" + Str(a1))
End Sub
```

Пример 5. Выполнить пример 4, другим способом, с помощью вспомогательных переменных.

Решение.

```
Sub выражение5()
Dim A, b, c, d, a1, a2, b1, c1, c2, x, y As Double
x = InputBox("Введите x=")
y = InputBox("Введите y=")
A = 2 * x - 3 * y
b = (2 * x - 3 * y) / 2
c = (2 * x - 3 * y) / 2 * x
d = (2 * x - 3 * y) / (2 * x)
a1 = (2 * x - 3 * y) / (2 * x) + (5 - x) / (3 + y)
‘ новое решение
b1 = A / 2
c1 = b * x
c2 = b / (2 * x)
a2 = d + (5 - x) / (3 + y)
MsgBox ("a=" + Str(A))
MsgBox ("b=" + Str(b))
MsgBox ("c=" + Str(c))
MsgBox ("d=" + Str(d))
MsgBox ("a1=" + Str(a1))
MsgBox ("b1=" + Str(b1))
MsgBox ("c1=" + Str(c1))
MsgBox ("c2=" + Str(c2))
MsgBox ("a2=" + Str(a2))
End Sub
```

Пример 6. Вычислить площадь треугольника по трем известным сторонам. Например, *a = 3, b = 4, c = 5*.

Решение.

```
Sub Герон1()
Dim A, b, c, p, s As Double
A = 3
b = 4
c = 5
p = (A + b + c) / 2
s = Sqr(p * (p - A) * (p - b) * (p - c))
MsgBox ("s=" + Str(s))
End Sub
```

Пояснение решения.

Для решения задачи используется формула Герона.

Пример 7. Вычислить площадь треугольника по трем известным сторонам.

Решение.

```
Sub Герон2()
Dim A, b, c, p, s As Double
A = Val(InputBox("Введите a="))
b = Val(InputBox("Введите b="))
c = Val(InputBox("Введите c="))
p = (A + b + c) / 2
s = Sqr(p * (p - A) * (p - b) * (p - c))
MsgBox ("s=" + Str(s))
```

End Sub

Пояснение решения.

Код Val(InputBox("Введите a=")) преобразует введенное значение через InputBox в число, так как InputBoxвозвращает строку. Если такого преобразования не сделать, то программа правильно вычислять *s* не будет.

Пример 8. Вычислить гипотенузу прямоугольного треугольника по двум катетам.

Решение.

```
Sub гипотенуза()
Dim a, b, c, p, s As Double
a = Val(InputBox("Введите a="))
b = Val(InputBox("Введите b="))
c = Sqr(a ^ 2 + b ^ 2)
MsgBox ("c=" + Str(c))
End Sub
```

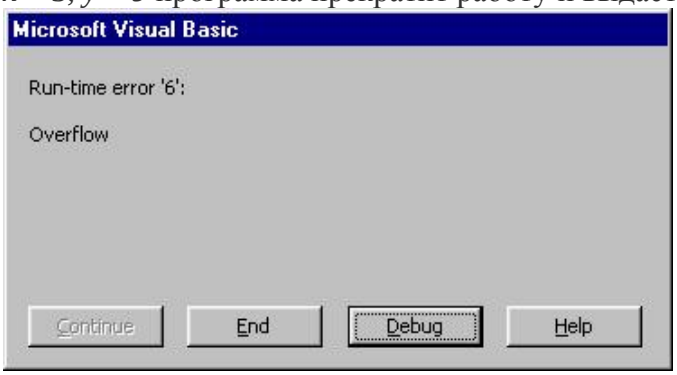
Организация ветвления на языке Visual Basic for Application

Необходимость обработки критических ситуаций возникает при программировании довольно часто.

Например, при решении задачи: вычислить значение выражения $\frac{x+y}{x-y}$.

```
Sub выражение6()
Dim x, y, z As Double
x = Val(InputBox("Введите x="))
y = Val(InputBox("Введите y="))
z = (x + y) / (x - y)
MsgBox ("z=" + Str(z))
End Sub
```

При вводе значений $x = 3$, $y = 3$ программа прекратит работу и выдаст сообщение



о прекращении работы, так как произошло деление на 0. Нажатие на кнопку ***Debug*** позволяет перейти в строку, в которой состоялась ошибка выполнения (*Run-time error*).

Необходимо, преобразовать программу следующим образом:

```
Sub выражение7()
Dim x, y, z As Double
x = Val(InputBox("Введите x="))
y = Val(InputBox("Введите y="))
If x - y <> 0 Then
z = (x + y) / (x - y)
MsgBox ("z=" + Str(z))
Else
```

```
MsgBox ("Знаменатель равен =0")
End If
End Sub
```

Для организации ветвлений в языке VBA предусмотрено несколько операторов: If и Select Case.

Общий вид оператора If:

If выражение **Then** [инструкции]
[**ElseIf** выражение-*n* **Then** [*иначе_если_инструкции*] ...
[**Else** [иначе_инструкции]]
End If

Выражение должно возвращать логическое значение: истина или ложь (*True* или *False*).

Общий вид оператора Select Case:

Select Case выражение
[**Case** выражение-*n*
[инструкции -*n*]] ...
[**Case Else**
[иначе_инструкции]]
End Select

Пример использования Select Case.

```
Sub пример_select_case()
    Dim Number
    Number = 8
    Select Case Number
    Case 1 To 5
    MsgBox "Между 1 и 5"
    Case 6, 7, 8
    MsgBox "Между 6 и 8"
    Case 9 To 10
    MsgBox "Между 9 и 10"
    Case Else
    MsgBox "Не в диапазоне от 1 до 10"
    End Select
End Sub
```

При программировании исключительных ситуаций необходимо предусмотреть все возможные случаи. Поясним это на примерах.

Пример 1. Вычислить значение выражения $\frac{x+y}{x-y}+\sqrt{x}$

Решение.

```
Sub выражение8()
Dim x, y, z As Double
x = Val(InputBox("Введите x="))
y = Val(InputBox("Введите y="))
If x - y <> 0 And x > 0 Then
z = (x + y) / (x - y) + Sqr(x)
MsgBox ("z=" + Str(z))
```

```
Else
MsgBox ("Выражение не имеет смысла")
End If
End Sub
```

Пояснение решения.

Исключительная ситуация возникает, когда в знаменателе получается нуль и подкоренное выражение меньше нуля.

Пример 2. Вычислить площадь треугольника по трем известным сторонам.

Решение.

```
Sub Герон3()
Dim A, b, c, p, s As Double
A = Val(InputBox("Введите a="))
b = Val(InputBox("Введите b="))
c = Val(InputBox("Введите c="))
If (A + b > c) And (A + c > b) And (b + c > A) Then
p = (A + b + c) / 2
s = Sqr(p * (p - A) * (p - b) * (p - c))
MsgBox ("s=" + Str(s))
Else
MsgBox ("Треугольник не существует")
End If
End Sub
```

Пояснение решения.

Предложенная программа проверяет существование треугольника, и не будет работать при введенных отрицательных значениях *a, b, c*.

Правильное решение в примере 3.

Пример 3. Вычислить площадь треугольника по трем известным сторонам.

```
Sub Герон4()
Dim A, b, c, p, s As Double
Dim d1, d2, tr_ok As Boolean
A = Val(InputBox("Введите a="))
b = Val(InputBox("Введите b="))
c = Val(InputBox("Введите c="))
d1 = (A >= 0) And (b >= 0) And (c >= 0)
d2 = (A + b > c) And (A + c > b) And (b + c > A)
tr_ok = d1 And d2
If tr_ok Then
p = (A + b + c) / 2
s = Sqr(p * (p - A) * (p - b) * (p - c))
MsgBox ("s=" + Str(s))
Else
MsgBox ("Треугольник не существует")
End If
End Sub
```

Пояснение решения.

Программа вычисляет площадь треугольника, правильно обрабатывая исключительные ситуации.

Программирование циклов на языке Visual Basic for Application

Организация выполнения повторяющихся действий в VBA может быть выполнена несколькими операторами, которые условно разделяют на цикл-пока, цикл-до, цикл-для.

Оператор While

Общий вид оператора While:

While выражение
[инструкции]
Wend

Оператор While предназначен для организации цикла-пока.

Инструкции будут выполняться пока *выражение* будет истинно.

Пример. Вычислить сумму чисел от 0 до 100.

Решение.

```
Sub сумма1()
Dim x, s As Double
x = 0
s = 0
While x <= 100
s = s + x
x = x + 1
Wend
MsgBox ("s=" + Str(s))
End Sub
```

Пояснение решения.

В переменной *s* накапливается значение суммы.

Оператор Do While

Общий вид оператора Do While:

Do [**While** выражение]
[инструкции]
[**Exit Do**]
[инструкции1]
Loop

Оператор Do While предназначен для организации цикла-пока.

Инструкции будут выполняться пока *выражение* будет истинно. Конструкция Exit Do предназначена для преждевременного выхода из цикла.

Пример. Вычислить сумму чисел от 0 до 100.

Решение.

```
Sub сумма2()
Dim x, s As Double
x = 0
s = 0
Do While x <= 100
s = s + x
x = x + 1
Loop
MsgBox ("s=" + Str(s))
End Sub
```

Оператор Do Loop Until

Общий вид оператора Do Loop Until:

Do
[инструкции]
[**Exit Do**]

[инструкции1]
Loop [**Until** выражение]
Оператор Do Loop Until предназначен для организации цикла-до.
Инструкции будут выполняться до момента, когда *выражение* станет истинным. Конструкция Exit Do предназначена для преждевременного выхода из цикла.
Пример. Вычислить сумму чисел от 0 до 100.
Решение.

```
Sub сумма3()
Dim x, s As Double
x = 0
s = 0
Do
s = s + x
x = x + 1
Loop Until x > 100
MsgBox ("s=" + Str(s))
End Sub
```

Оператор For

Общий вид оператора For:
For счетчик = начальное_знач **To** конечное_знач [**Step** шаг]
[инструкции]
[**Exit For**]
[инструкции1]
Next [счетчик]
Оператор For предназначен для организации цикла-для.
Инструкции будут выполняться определенное количество раз, задаваемое в счетчик, начиная с начального значение (начальное_знач) до конечного значения (конечное_знач) с некоторым шагом (шаг). Конструкция Exit Forпредназначена для преждевременного выхода из цикла.
Пример. Вычислить сумму чисел от 0 до 100.
Решение.

```
Sub сумма4()
Dim x, s As Integer
s = 0
For x = 0 To 100
s = s + x
Next x
MsgBox ("s=" + Str(s))
End Sub
```

Примеры решения задач

Пример 1. Представить таблицу квадратов чисел от 2 до 10.
Решение.
Для отображения результатов воспользуемся Excel и для доступа к ячейкам функцией Cells (номер_строки, номер_колонки).

```
Sub квадраты()
Dim x
Cells(2, 1).Value = "x"
Cells(3, 1).Value = "x^2"
For x = 2 To 10
Cells(2, x).Value = x
```

```
Cells(3, x).Value = x ^ 2
Next x
End Sub
```

	A	B	C	D	E	F	G	H	I	J
1										
2	x	2	3	4	5	6	7	8	9	10
3	x^2	4	9	16	25	36	49	64	81	100

Пример 2. Составить программу определения наибольшего общего делителя (НОД) двух натуральных чисел.

Решение.

Наибольший общий делитель двух натуральных чисел — это самое большое натуральное число, на которое они делятся. Например, у чисел 12 и 18 наибольшие делители: 2, 3, 6. наибольшим общим делителем является число 6. Это записывается так:

НОД(12, 18) = 6.

Идея *алгоритма Евклида* для нахождения НОД основана на том свойстве, что если M>N, то

НОД(M, N) = НОД(M-N, N).

Иначе говоря, НОД двух натуральных чисел равен НОД их положительной разности и меньшего числа.

```
Sub Евклид()
Dim M, N, NOD
M = Cells(1, 2)
N = Cells(2, 2)
While M <> N
If M > N Then
M = M - N
Else
N = N - M
End If
Wend
NOD = M
Cells(3, 2).Value = NOD
End Sub
```

	A	B
1	M=	12
2	N=	18
3	НОД=	6

Пример 3. Построить график функции: улитку Паскаля.

Улитка Паскаля задается следующим образом:

x=A*cos(t)+B*cos(t)

y=A*cos(t)sin(t)+B*sin(t), A>B, B>0, 0<=t<2*Pi

Решение.

1. Подготовить данные в электронной таблице

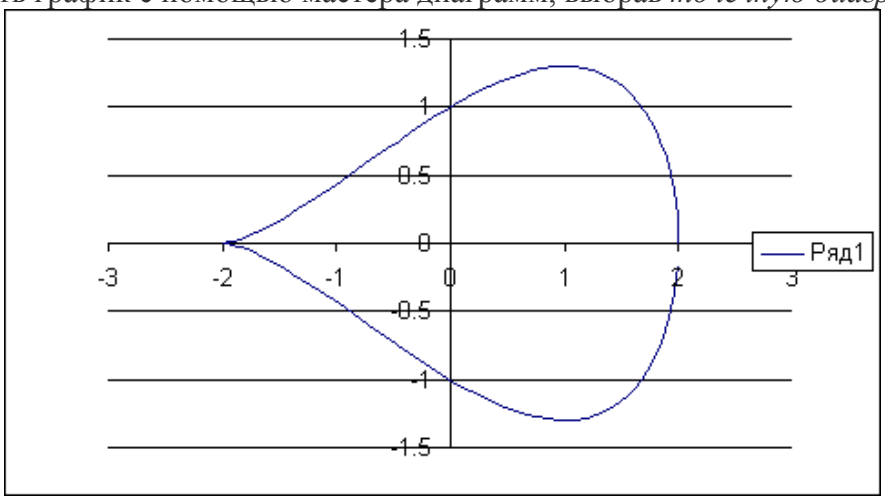
	A	B
1	a=	1
2	b=	1
3	x=	
4	y=	

2. Ввести код программы.

```
Sub улитка_паскаля()
Dim a, b, Pi, t As Double
Dim i As Integer
```

```
a = Cells(1, 2)
b = Cells(2, 2)
Pi = 3.14
i = 2
t = 0
While t <= 2 * Pi
x = a * Cos(t) + b * Cos(t)
y = a * Cos(t) * Sin(t) + b * Sin(t)
Cells(3, i).Value = x
Cells(4, i).Value = y
t = t + 0.1
i = i + 1
Wend
End Sub
```

3. Построить график с помощью мастера диаграмм, выбрав *точечную диаграмму*.

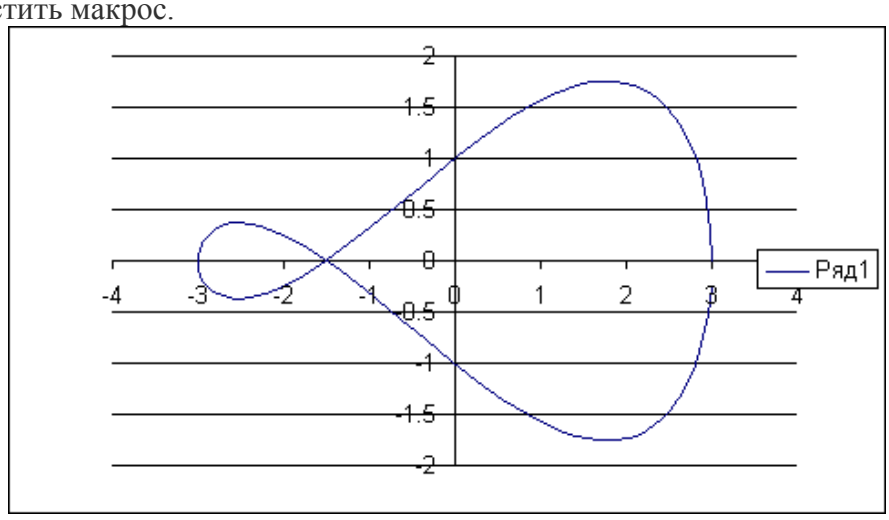


4. Изменить данные *a* и *b*.

A=2

B=1

5. Перезапустить макрос.



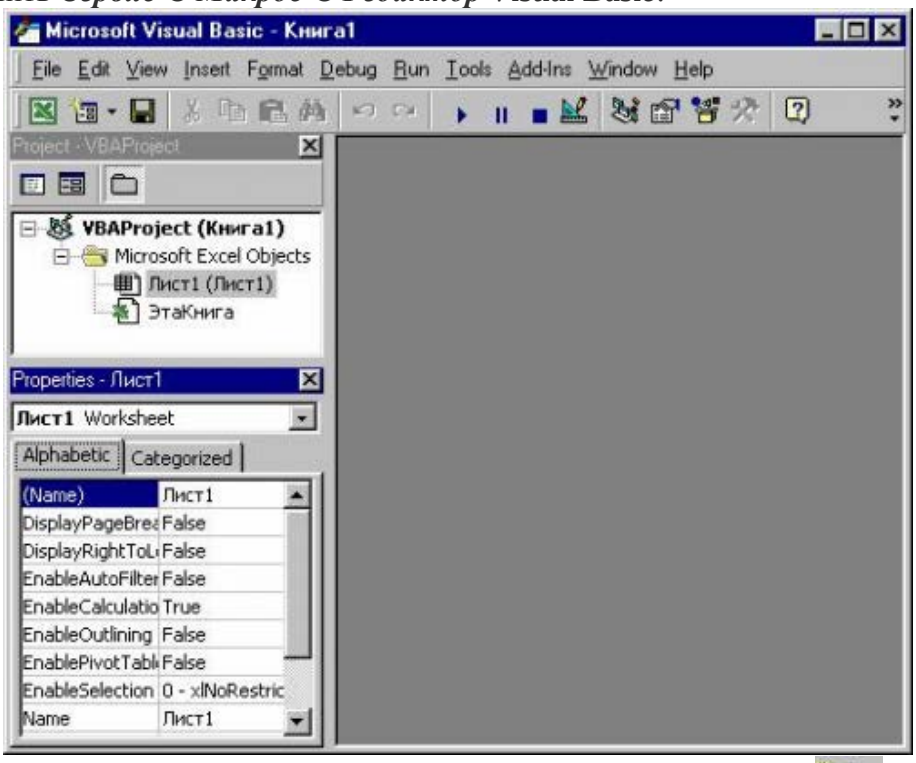
Создание пользовательских форм на языке Visual Basic for Application

Для пользователя часто необходимо создать собственную среду, в которой на специальной форме располагаются все необходимые инструменты для работы. VBA располагает для этого всеми средствами.

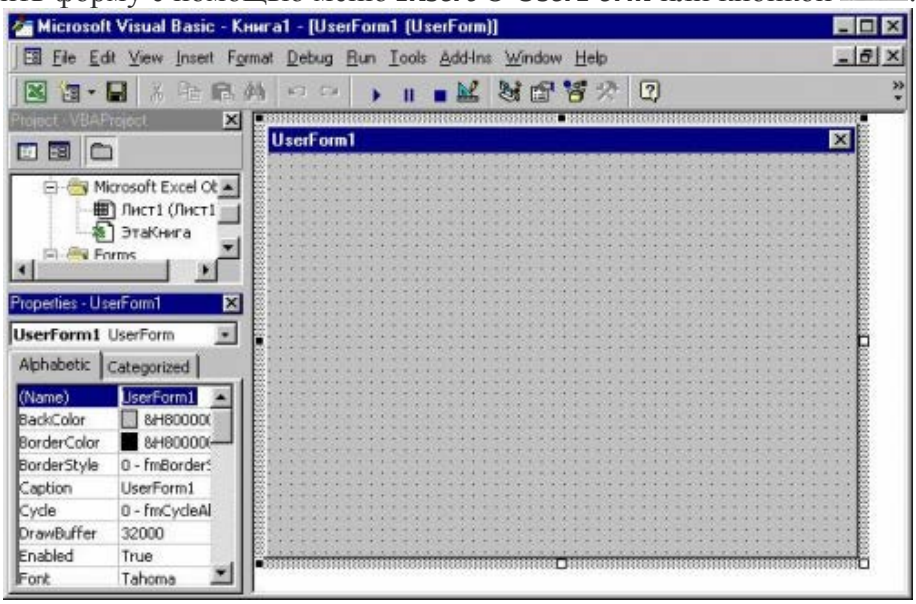
Приведем создание проекта «Вычисление площади треугольника по трем известным сторонам» по шагам.

Шаг 1. Создание первоначальной формы.

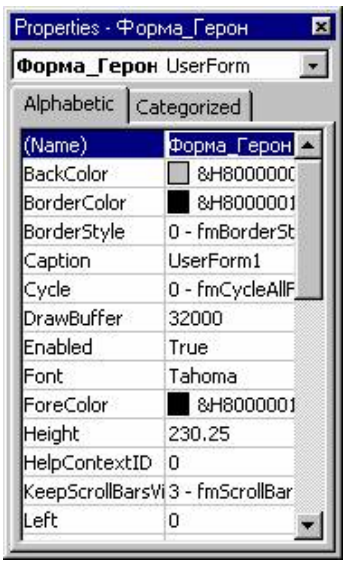
1.1. Выполнить ***Сервис ® Макрос ® Редактор Visual Basic***.



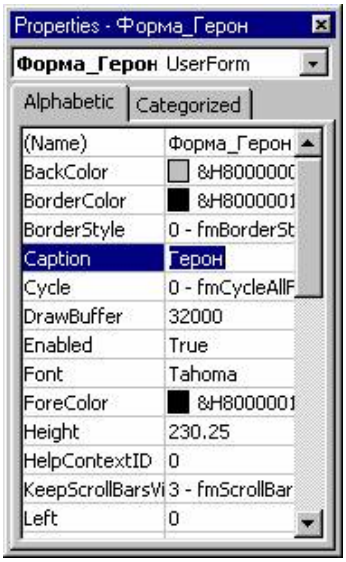
1.2. Вставить форму с помощью меню ***Insert ® UserForm*** или кнопкой  .



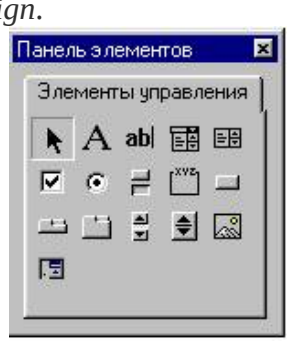
1.3. Дать внутреннее имя форме **Форма_Герон** с помощью окна Properties, выставляя свойство *(Name)* равноеФорма_Герон.



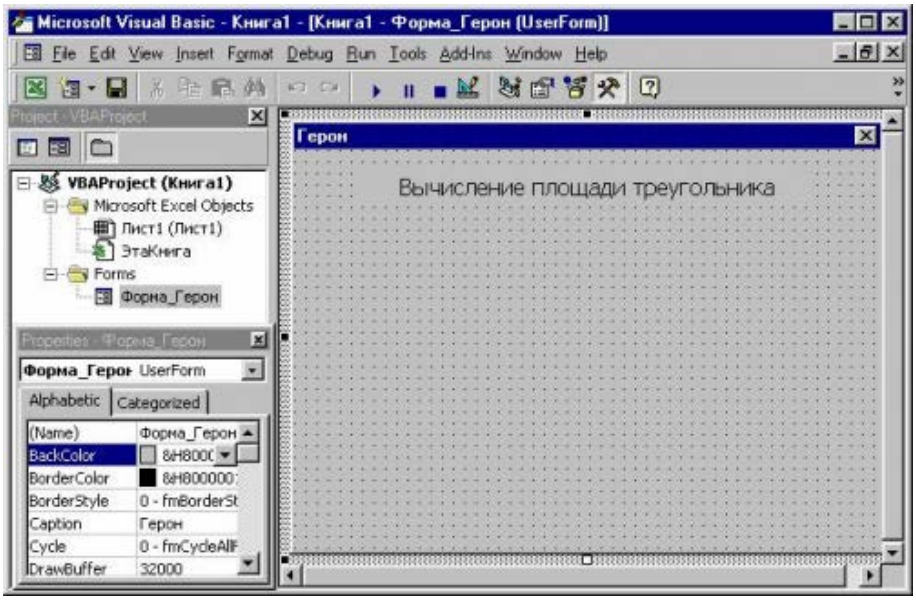
1.4. Присвоить заголовку форму имя **Герон** с помощью окна *Properties*, выставляя свойство *Caption* равное Герон.



1.5. Создать надпись «Вычисление площади треугольника» с помощью инструмента *Надпись*  панели элементов. При необходимости воспользоваться окном *Properties*, свойствами *Font*, *Align*.



1.6. В результате получиться следующий рисунок.

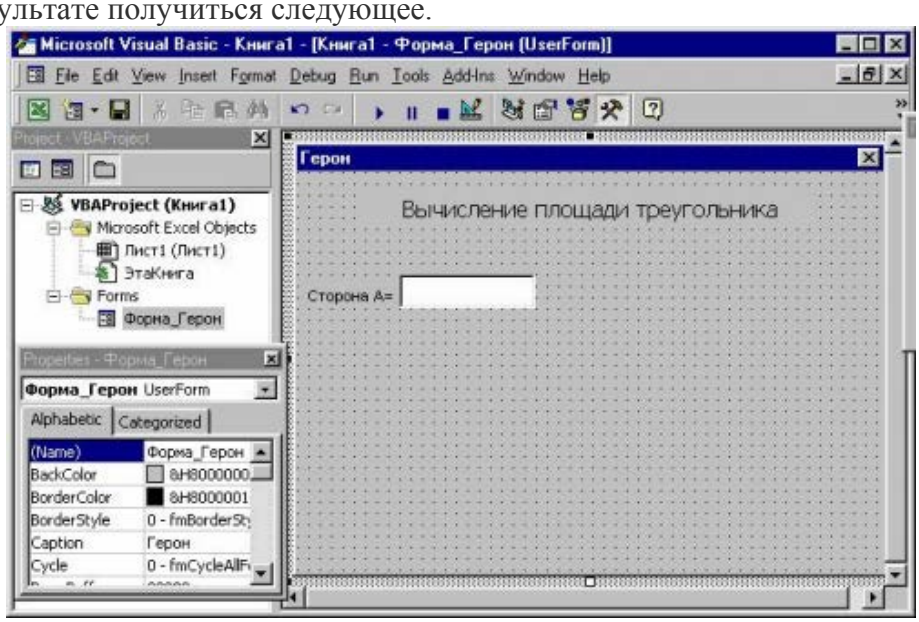


Шаг 2. Создание элементов для ввода данных.

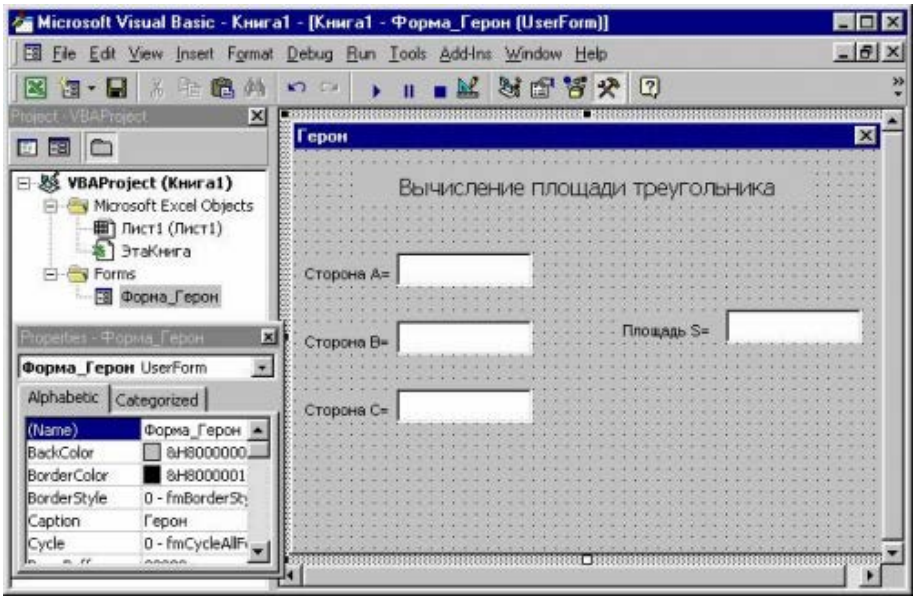
2.1. Создать блок для ввода данных стороны *a* с помощью инструмента *Поле* ab| панели элементов. В окне*Properties*, присвоить ему свойство *(Name)* равное a_Textbox.

2.2. Создать надпись «Сторона A=» с помощью инструмента *Надпись* A панели элементов. При необходимости воспользоваться окном *Properties*, свойствами *Font, Align*.

2.3. В результате получиться следующее.

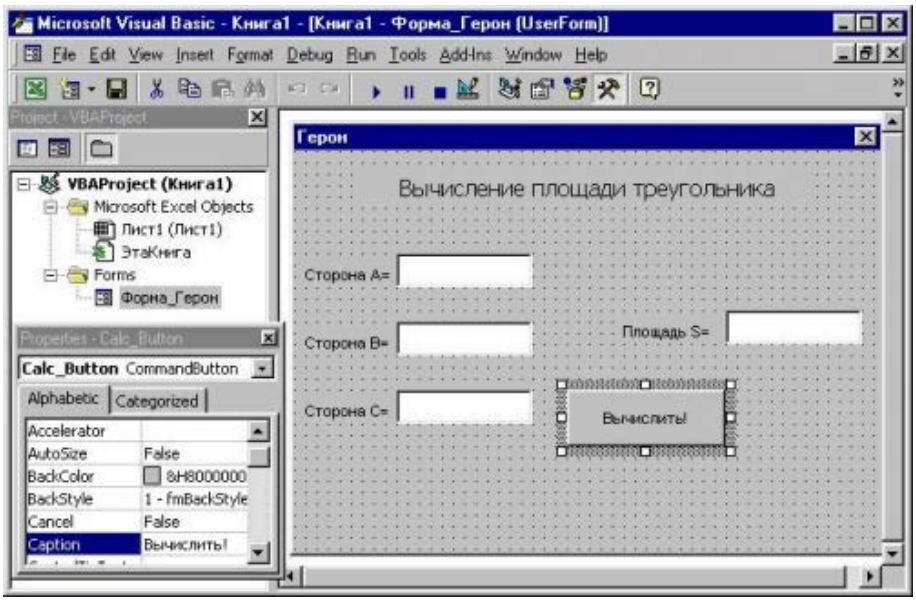


2.4. Аналогично создать блоки для ввода значений сторон B, C и вывода площади треугольника S. Для B —b_Textbox, C — c_Textbox, S — s_Textbox.



Шаг 3. Создание элемента для начала вычислений.

3.1. Создать кнопку «Вычислить!» с помощью инструмента *Кнопка*  панели элементов. В окне *Properties*, выставить свойства *(Name)* равное Calc_Button, *Caption* равное «Вычислить!».



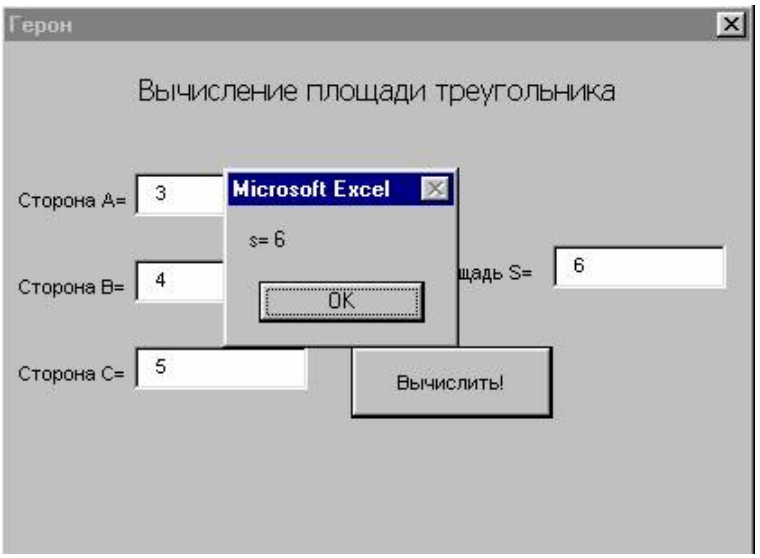
3.2. Сделать двойной щелчок на кнопке ***Вычислить!*** и ввести программный код.

```
Private Sub Calc_Button_Click()
Dim A, b, c, p, s As Double
Dim d1, d2, tr_ok As Boolean
A = Val(a_TextBox.Value)
b = Val(b_TextBox.Value)
c = Val(c_TextBox.Value)
d1 = (A >= 0) And (b >= 0) And (c >= 0)
d2 = (A + b > c) And (A + c > b) And (b + c > A)
tr_ok = d1 And d2
If tr_ok Then
p = (A + b + c) / 2
s = Sqr(p * (p - A) * (p - b) * (p - c))
s_TextBox.Value = s
MsgBox ("s=" + Str(s))
```

```
Else
MsgBox ("Треугольник не существует")
s_TextBox.Value = "Треугольник не существует"
End If
End Sub
```

Шаг 4. Запуск формы.

4.1. Осуществить запуск формы с помощью ***Run ® Run Sub/UserForm*** или с помощью кнопки .



Обработка массивов на языкеVisual Basic for Application

В повседневной жизни часто приходится встречаться с информацией, представленной в табличном виде. Для обработки можно использовать электронные таблицы, а также VBA.

Массив – это структурированный тип данных, конечная упорядоченная совокупность данных одного типа, доступ к которым осуществляется по индексу (порядковому номеру). Массивы необходимы, когда требуется несколько раз обращаться к одной и той же группе однотипных данных.

Массивы могут содержать данные любого типа: тип элементов массива распознается по идентификатору. Массивы необходимо объявлять. С помощью оператора Dim. При объявлении указывается имя массива, размерность и количество элементов по каждой размерности (эти количества должны быть определены до объявления массива).

Использование массивов значительно упрощает работу с группами однотипных данных.

Все действия с массивами выполняются поэлементно, в цикле. Поскольку массив — это последовательность с известным числом элементов, удобнее использовать цикл For.

Пример. Вычислить средний рост по данным, записанным в электронной таблице.

	A	B
1	**Фамилия**	**Рост (см)**
2	Иванов	140
3	Петров	142
4	Сидоров	135
5	Федоров	139
6	Розина	143
7	Николаева	139
8		
9	**Средний рост**	
10		

Результат вычисления среднего роста будут записаны в ячейку B9.

Решение.

```
Sub средний_рост()
Dim Rost(6) As Double
Dim i As Integer
Dim Сумма, Среднее As Double
' ввод таблицы для обработки
For i = 1 To 6
Rost(i) = Cells(1 + i, 2).Value
Next i
' нахождение суммы чисел в таблице
Сумма = 0
For i = 1 To 6
Сумма = Сумма + Rost(i)
Next i
' вычисление среднего
Среднее = Сумма / 6
'вывод
Cells(9, 2).Value = Среднее
MsgBox (Среднее)
End Sub
```

Пояснение решения.

В строке Dim Rost(6) As Double объявляется массив чисел двойной точности именем *Rost* размерностью 6, то есть одномерная таблица *Rost* емкостью 6 (шесть) ячеек.

Аналогичным образом можно обрабатывать и двумерные массивы.

Пример. Определить, является ли данный квадратный массив симметричным относительно своей главной диагонали. Данные записаны в электронной таблице.

Решение.

```
Sub simetria()
Const n = 4
Dim i, j
Dim x(n, n)
Dim t, check As Boolean
For i = 1 To n
For j = 1 To n
x(i, j) = Cells(i, j)
Next
Next
t = True 'предположим, что матрица симметрична
i = 2
While t And (i < n)
```

```
j = 1
While (j < i) And (x(i, j) = x(j, i))
j = j + 1
Wend
t = (j = i)
i = i + 1
Wend
check = t
MsgBox check
End Sub
```

Обработка строк на языкеVisual Basic for Application

Строка — это последовательность символов. Строковые величины могут быть переменными или константами. Символы, заключенные в кавычки, являются строками.

В VBA существует несколько функций для обработки строк.

Функция Mid

Общий вид функции Mid:

Mid(Строка, Начальная_позиция[, Длина])

Функция Mid возвращает вырезку из строки *Строка*, начиная со позиции *Начальная_позиция*, длиною *Длина*.

Пример 1.

```
Sub пример_mid()
Dim MyString, Word1, Word2, Word3
MyString = "Демо функции Mid"
Word1 = Mid(MyString, 1, 4) ' результат "Демо"
MsgBox Word1
Word2 = Mid(MyString, 14, 3) ' результат "Mid"
MsgBox Word2
Word3 = Mid(MyString, 6)   ' результат "функции Mid"
MsgBox Word3
End Sub
```

Функция Len

Общий вид функции Len:

Len (Строка)

Функция Len возвращает длину строки *Строка*.

Пример 2.

```
Sub пример_len()
Dim MyString
Dim Длина
MyString = "Демо функции Len"
Длина = Len(MyString)
MsgBox Длина
End Sub
```

Функция InStr

Общий вид функции InStr:

InStr([нач_позиция,]Строка1, Строка2[, Опция_1_или_0])

Функция InStr номер первого вхождения в строке *Строка1* строки *Строка2*, начиная с позиции *Нач_позиция*.

Пример 3.

```
Sub пример_instr()
Dim Mystring
Mystring = "Где Вася"
номер = InStr(Mystring, "Вася") ' результат 5
MsgBox номер
End Sub
```

Кроме приведенных функций в VBA имеются: Left (вырезка слева), Right (вырезка справа), Trim (убирает пробелы слева и справа), StrComp (сравнение строк) и др.

Приведем пример обработки строк.

Пример 4. В строке подсчитать количество цифр.

Решение.

```
Sub пример_str_4()
Dim Mystring, char
Dim i, n
Mystring = InputBox("Введите строку")
n = 0
For i = 1 To Len(Mystring)
char = Mid(Mystring, i, 1)
If char >= "0" And char <= "9" Then n = n + 1
Next i
MsgBox n
End Sub
```

По материалам сайта itteach.ru