Лекция 5

На предыдущей лекции мы рассмотрели вопросы реализации разветвляющихся функций средствами программы ***Excel***. Были записаны реализация функции с двумя ветвями и реализация функции с тремя ветвями. Особое внимание мы уделили вопросу разработки блок-схем алгоритмов.

Сегодня мы рассмотрим, как функции такого вида могут быть реализованы средствами ***Visual Basic*** (***VB***).

1. Реализация разветвляющейся функции средствами Visual Basic'a

Функция пользователя, как мы уже видели, может быть реализована в среде ***VB*** в результате выполнения последовательности шагов, первым из которых будет переход в среду ***VB:***

- в меню "***Сервис***" программы ***Excel*** выбираем команду "***Макрос ►***", а затем в открывающемся подменю команду "***Редактор Visual Basic";***
- в открывающемся окне редактора ***VB*** для редактирования открываем окно стандартного модуля, в котором находятся записанные ранее функции пользователя. Для этого в окне ***Project*** на дереве структуры выбираем и открываем папку ***"Modules"***;
- в этой папке выбираем и открываем модуль, в который ранее были записаны функции, реализующие отдельные ветви разветвляющейся функции;
- для реализации разветвляющейся функции создадим новую функцию: в меню ***VB*** выбираем команду ***"Insert"***. В открывающемся выпадающем меню выбираем строку "***Procedure…***". Процесс создания функции пользователя мы уже рассматривали. (*Лекция 3*).
- в открывающемся при выборе этой команды ДО "***Add Procedure***" в поле ввода "***Name:***" ("*Имя:*"), записываем имя создаваемой функции, например ***FVB***. В поле "***Type***" включаем переключатель "***Function***" (*Функция*) и нажимаем кнопку ***OK***.

- В окне редактора ***VB*** появляется заготовка текста функции ***FVB,*** содержащая две строки:

-строку заголовка функции

Public Function FVB()

и строку завершения функции

End Function

Между этими строками мы должны поместить операторы, реализующие алгоритм вычисления значения разветвляющейся функции. В языке ***Visual Basic*** для реализации блока ***"Решение"*** используется оператор условного перехода

If <логическое_выражение> Then <Оператор, если True>

Else < Оператор, если False>.

Ключевые слова, используемые в этом операторе ***If*** , переводится как "*Если*", ***Then*** , переводится как "*Тогда*" и ***Else*** переводится как "*Иначе*" ("*В противном случае*")

Аналогично функции **ЕСЛИ** в Excel оператор условного перехода ***If … Then … Else*** допускает вложение одного условия в другое, поэтому для реализации алгоритма, приведённого на рис. 4.6 необходимо использовать два уровня оператора ***If … Then … Else***.

На первом уровне проверяется, как и при записи функции **ЕСЛИ**, принадлежность аргумента области от (-∞) до α. Если это условие выполнено, т.е. его значение логического выражения, записанного после ***If,*** равно **TRUE** (ИСТИНА), то выполняется оператор, находящийся после ключевого слова ***Then.***

Если же значение логического выражения, записанного после ***If,*** равно **FALSE** (ЛОЖЬ), то выполняется оператор, находящийся после ключевого слова ***Else.***

Запишем реализацию функции пользователя с именем ***FVB(),*** обеспечивающую вычисление значений разветвляющейся функции (4.4)

$$F(x) = \begin{cases} sin(a*x) \quad при \quad x \le \alpha, \\ a*x + b \quad при \quad \alpha < x \le \beta, \\ ctg(a*x) \quad при \quad \beta < x. \end{cases} \qquad (4.4)$$

На примере записи этой функции познакомимся с основными положениями языка и с некоторыми правилами записи процедур в языке ***Visual Basic.***

2. Алфавит языка программирования Visual Basic

Для записи всех объектов, Объектами, значения которых передаются или определяются в тексте процедур, могут быть константы, переменные, функции, процедуры и др.

Записывая код программы, программист определяет имена всех используемых в ней объектов. Такие имена называются **идентификаторами**.

В ***VBA*** определены следующие *правила записи идентификаторов*:

- идентификатор объекта ***VBA*** может содержать любую комбинацию букв латинского алфавита, цифр и некоторых специальных символов, начинающуюся с буквы;
- в идентификаторах нельзя использовать пробелы, точки (**.**), запятые (**,**), восклицательные знаки (**!**), а также символы **@, &, $** и #;
- максимальная длина идентификатора 255 символов;
- не следует использовать в программе идентификаторы, совпадающие с ключевыми словами ***VBA,*** именами встроенных функций, процедур ***VBA;***
- идентификаторы должны быть уникальны в пределах области определения.

В ***VBA*** не различаются строчные и заглавные буквы. Включение в идентификатор заглавных букв используется для облегчения понимания назначения и содержания констант, переменных, массивов и других именуемых разработчиками программ объектов ***VBA***.

Так запись идентификатора ***ChisloStudentov***, позволяет, прочитав этот идентификатор определить его назначение. Такая запись более информативна, чем запись этого же идентификатора строчными или заглавными буквами.

В записи программ, функций и процедур идентификатор переменной используется для обозначения некоторого информационного объекта. Идентификаторы в ***VBA*** могут вводиться в текст по мере необходимости, без каких-либо предварительных объявлений и описаний.

Эта возможность и использована в приводимом ниже тексте функции.

Рассмотрим и проанализируем построчно приводимый ниже текст функции ***FVB,*** записанный с минимальными знаниями языка ***VBA.***

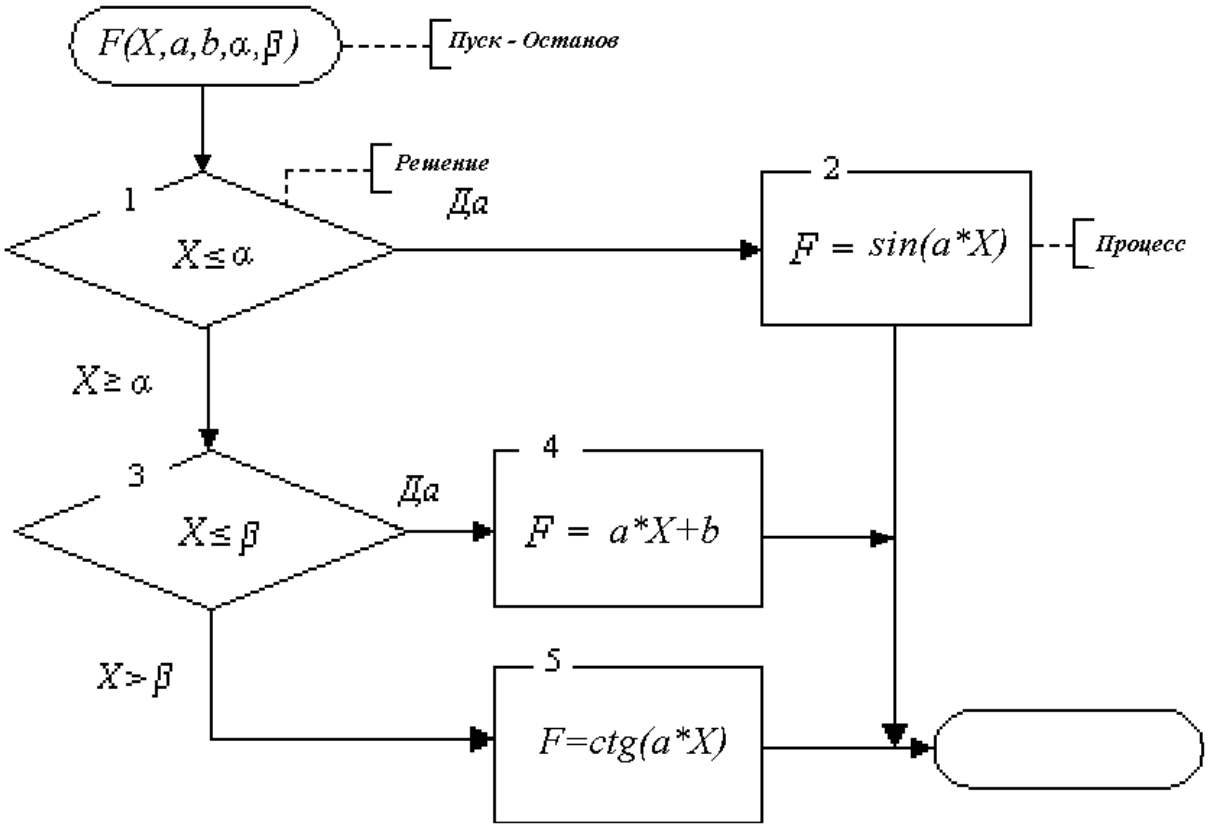


Рис. 4.6. Блок-схема алгоритма вычисления значения разветвляющейся функции ***F(x)***

```
Public Function FVB(X, a, b, alfa, betta)                    (1)
'X – аргумент функции, a,b,alfa,betta - параметры            (2)
If (X <= alfa) Then FVB = Sin(X*a) _                         (3)
Else _                                                       (4)
    If (X <= betta) Then FVB = a * X + b _                   (5)
    Else FVB = 1 / Tan(a * X)                                (6)
End Function                                                 (7)
```

1) В созданный ранее заголовок функции, мы ввели **список параметров** - имён объектов: ***(X, a, b, alfa, betta)***. Значения параметров должны быть определены в обращении к функции.

   В списке параметров ***X*** – аргумент функции, ***a*** и ***b*** определяют конкретную реализацию функции ***F(X)***, ***alfa*** и ***betta,*** определяют разбиение интервала изменения аргумента ***X*** на ветви.

2) Вторая строка программного кода начинается **с символа (') – апостроф.** Синтаксис ***VBA*** определяет, что апостроф, стоящий в строке программы, определяет начало **комментария**, не анализируемого компилятором ***VBA.*** Комментарии, включаемые в тексты программ, процедур и функций используются для пояснений, облегчающих понимание действий, выполняемых программным кодом. Для определения того что данная строка в тексте программы является комментарием можно также использовать ключевое слово ***Rem*** (от *Remark* - замечание, комментарий).

3)-6) строки программного кода являются записью двухуровневого условного оператора ***If … Then … Else.***

   Код третьей строки обеспечивает реализацию блоков (1) и (2) алгоритма, приведённого на рис. 4.6.

   Пятая и шестая строки реализуют проверку второго условия и вычисления для второй и третьей ветвей функции, т.е. действия определяемые блоками (3), (4) и (5) алгоритма.

Следует отметить, что строки оканчиваются символами **<пробел>** и **<подчёркивание>**. Эти два символа, записанные в конце строки определяют следующую строку программы как продолжение этой строки. Таким образом реализуется возможность записи ***многострочных операторов.***

Результат вычислений определяет значение имени функции ***FVB.***

3. Определение и представление данных в VBA

Информационные объекты записываются и хранятся в ячейках памяти ЭВМ. Определив ***идентификатор (имя)*** некоторого объекта в тексте программы мы тем самым определили, что для данного объекта в памяти ЭВМ компилятор должен выделить некоторую область, достаточную для размещения объекта.

VBA - объектно-ориентированный язык программирования. **Объект** – это информационная структура, объединяющая в себе как некоторые информационные элементы, так и методы, позволяющие выполнять операции над этими информационными элементами.

Базовыми информационными объектами VBA, определяющими самый простой уровень информационных структур **VBA**, являются ***константы и переменные***.

Способ представления констант и переменных определяется их типом, определяемым при записи их в память ЭВМ как ***тип данных***.

Тип данных определяет **множество допустимых значений**, которое может принимать информация, определяемая этим типом.

В языке ***Visual Basic*** определены следующие типы данных:

Таблица 1.

№	Тип	Описание	Диапазон	Размер
1	***Byte***	Байт	0 - 255	8 bit (1байт)
2	***Boolean***	Логический	True - False	16-bit (2-byte)
3	***Integer***	Целое	-32 768 – 32 767	16-bit (2-byte)
4	***Long***	Длинное целое	-2 147 483 648 – 2 147 483 647	32-bit (4-byte)
5	***Single***	С плавающей точкой обычной точности	±3.402823E38 - ±1.401298E-45	32-bit (4-byte)
6	***Double***	С плавающей точкой двойной точности	±1.79769313486231E308 ±4.94065645841247E-324	64-bit (8-byte)
7	***Currency***	Денежный	-922,337,203,685,477.5808 922,337,203,685,477.5807	64-bit (8-byte)
8	***Decimal***	Масштабируемое целое	±79 228 162 514 264 337 593 543 950 33 5 ±0.0000000000000000000000000001	96-bit (12-byte)
9	***Date***	Дата и время		64-bit (8-byte)
10	***Object***	Объект		
11	***String***	Строка переменной длины	2^{31} characters	
12	***String***	Строка постоянной длины	64K (2^{16}) characters	
13	***Variant***	Числовые подтипы		
14	***Variant***	Строковые подтипы		
15	***Type***	Типы данных, определяемые пользователем		

В памяти ЭВМ объект каждого типа занимает некоторое число байтов, обеспечивающее возможность записи любого из возможных для данного типа значений. Данные, используемые в текстах программ на языке VB, могут быть объявлены явно, средствами специальных *инструкций объявления типа переменных.*

4. Объявление констант

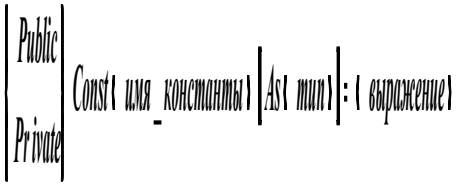
Константа – это именованный информационный объект, сохраняющий неизменяемое в процессе выполнения программы значение. Константа может быть строкой, числом, или любым выражением, содержащим последовательность числовых или логических операндов и операций кроме операции **Is** и операций возведения в степень.

В каждом приложении может быть определено свое множество констант. Необходимые константы могут быть определены пользователем с помощью оператора **Const.**

Константы могут быть определены в любом месте создаваемого кода.

Объявляя константу мы **задаём имя**, которому присвоено некоторое неизменяемое при выполнении программы значение. Это имя определяет ***именованную константу***.

Для реализации именованных констант в текст программы должна быть вставлена инструкция следующего формата:



Ключевое слово ***Public***, предшествующее оператору ***Const,*** определяет модуль, как область действия объявленной в этом модуле переменной.

Область действия констант, объявленных как ***Private***, ограничивается пределами структуры, в которой она объявлена.

Тип константы, может быть любым допустимым данных, кроме типа ***Object.*** Для каждой объявляемой константы необходимо, но не обязательно, использовать отдельную последовательность ***As <mun>***.

Примеры объявления констант:

Public Const conAge As Integer = 34, conPrice As Currency - 8000 – объявляется целочисленные константы ***conAge*** и ***conPrice.*** Общий префикс ***con*** используется в записи идентификаторов для выделения констант в тексте программы.

Значение константы ***conAge*** целочисленное и равно 34. Значение константы ***conPrice*** имеет тип ***Денежный*** и равно 8000.

По умолчанию область определения констант имеет тип ***Private***. В одно объявление может входить несколько констант.

5. Объявление переменных

Переменная – именованная область памяти, в которой может храниться информация, изменяемая в процессе выполнения программы. Имя каждой переменной уникально в пределах её области определения. Тип данных может быть определён (объявлен) или не указан.

В рассмотренной нами программе тип переменных, используемых в тексте программы, не объявляется явно, а определяется компилятором ***VB*** при трансляции программы. Такой способ неявного объявления переменных может в ряде случаев привести к возникновению трудно определяемых ошибок.

Примером возникновения такой ошибки может быть замена или пропуск одной буквы при записи идентификатора в программе (**ROTATION** и **ROTACION**). В этом случае компилятор не заметит ошибки и создаст в памяти ЭВМ две разные переменные. Значение одной из переменных окажется не определённым.

Более предпочтительным при записи программных текстов является вариант с включением в текст программы явных описаний всех

идентификаторов, определяемых программистом. Для того чтобы определить режим компиляции программы, в котором не допускается использование не объявленных заранее имён, в начале программного модуля следует поместить оператор ***Option Explicit***.

Наличие этого оператора в тексте программного модуля исключает использование в программе имён без их предварительного объявления.

Для объявления переменных в ***VBA*** используется оператор ***Dim*** (от ***dimension*** – размерность). Синтаксис этого оператора определяется как:

Dim varname1 [As [New] type][, varname2 [As [New] type]]...

где:

- ***varname1, varname2, ...*** **–** имена объявляемых переменных,
- ***As*** ключевое слово "*Как*", используемое для перехода к указанию типа объявляемой переменной ***type.***
- ***New*** ключевое слово "***Новый***", подразумевающее неявное создание объектов.
- ***type*** **-** необязательный элемент описания, определяющий ***тип данных для объявляемых переменных. (см. табл. 1).*** Для каждой объявляемой переменной необходимо использовать отдельную последовательность ***As type.***

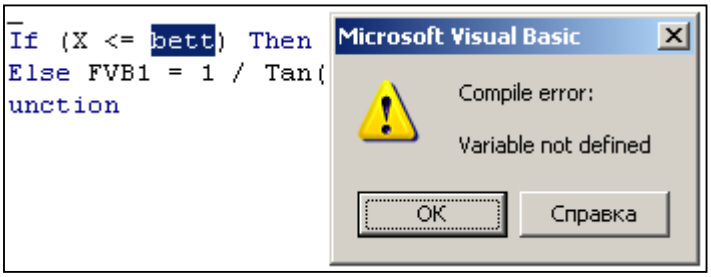
Создадим новый модуль, в первой строке которого запишем оператор ***Option Explicit***. А далее запишем текст функции ***FVB1(X, a, b, alfa, betta),*** учитывая всё вышесказанное.

```
Public Function FVB1(X As Double, a As Double, b As Double,
                     alfa As Double, betta As Double) As Double
'x – аргумент функции, a,b,alfa,betta – параметры функции
If (X <= alfa) Then FVB1 = Sin(X * a) _
Else _
   If (X <= bett) Then FVB1 = a * X + b _  ' ошибка при вводе имени
   Else FVB1 = 1 / Tan(a * X)
End Function
```

Для того чтобы проверить эффект, определяемый включением оператора ***Option Explicit*** в текст модуля, при записи функции допущена ошибка в записи имени параметра ***betta***. При попытке выполнить вызов

функции ***FVB1*** в окне редактора ***VBA*** будет выведено окно сообщения, показанное на рис. 5.1.

Рис. 5.1.



6. *Встроенные математические функции VBA*

При записи функций пользователя мы можем использовать различные функции, встроенные в ***VBA.*** В частности набор встроенных математических функций. В этот набор входят функции:

- **Abs(X)** – модуль числа ***X***;
- **Atn(X)** – арктангенс ***X***;
- **Cos(X)** – косинус ***X***;
- **Exp(X)** – экспонента числа ***X***, т.е. результат возведения числа ***e≈2.718282*** в степень ***X***;
- ***Fix(X), Int(X)*** – функции, возвращающие целую часть числа. Различие между этими функциями в том, что при отрицательных значениях аргумента ***X*** функция ***Fix*** возвращает ближайшее целое отрицательное число большее либо равное ***X***, а функция ***Int*** возвращает целое отрицательное, меньшее либо равное ***X***.
- ***Log(X)*** – натуральный логарифм ***X***;
- ***Rnd(X)*** – генератор случайных чисел;
- ***Sgn(X)*** – знак ***X***;
- ***Sin(X)*** – синус ***X***;
- ***Sqr(X)*** – корень квадратный из ***X***;
- ***Tan(X)*** – тангенс ***X***.

Derived Math Functions – производные математические функции

Function	Derived equivalents
Secant	Sec(X) = 1 / Cos(X)
Cosecant	Cosec(X) = 1 / Sin(X)
Cotangent	Cotan(X) = 1 / Tan(X)
Inverse Sine	Arcsin(X) = Atn(X / Sqr(-X * X + 1))
Inverse Cosine	Arccos(X) = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
Inverse Secant	Arcsec(X) = Atn(X / Sqr(X * X – 1)) + Sgn((X) – 1) * (2 * Atn(1))
Inverse Cosecant	Arccosec(X) = Atn(X / Sqr(X * X - 1)) + (Sgn(X) – 1) * (2 * Atn(1))
Inverse Cotangent	Arccotan(X) = Atn(X) + 2 * Atn(1)
Hyperbolic Sine	HSin(X) = (Exp(X) – Exp(-X)) / 2
Hyperbolic Cosine	HCos(X) = (Exp(X) + Exp(-X)) / 2
Hyperbolic Tangent	HTan(X) = (Exp(X) – Exp(-X)) / (Exp(X) + Exp(-X))
Hyperbolic Secant	HSec(X) = 2 / (Exp(X) + Exp(-X))
Hyperbolic Cosecant	HCosec(X) = 2 / (Exp(X) – Exp(-X))
Hyperbolic Cotangent	HCotan(X) = (Exp(X) + Exp(-X)) / (Exp(X) – Exp(-X))
Inverse Hyperbolic Sine	HArcsin(X) = Log(X + Sqr(X * X + 1))
Inverse Hyperbolic Cosine	HArccos(X) = Log(X + Sqr(X * X – 1))
Inverse Hyperbolic Tangent	HArctan(X) = Log((1 + X) / (1 – X)) / 2
Inverse Hyperbolic Secant	HArcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)
Inverse Hyperbolic Cosecant	HArccosec(X) = Log((Sgn(X) * Sqr(X * X + 1) + 1) / X)
Inverse Hyperbolic Cotangent	HArccotan(X) = Log((X + 1) / (X – 1)) / 2
Logarithm to base N	LogN(X) = Log(X) / Log(N)

Значение π

В ***VBA*** нет функции, возвращающей числовое значение константы π. Поэтому определить значение константы π можно двумя способами. Во-первых, использовав объявление ***Pi,*** как константы ***Pi*** = 3.1415926535897932, либо определить ***Pi*** как значение функции ***Pi = 4*Atn(1).***

Ещё один способ, это включение в инструкцию VBA обращения Application.Pi(), обеспечивающего возврат значения π.

Например, функция

```
Public Function PiQ(X)
    PiQ = Application.Pi() * X
```

End Function

Обеспечивает вычисление значения ***PiQ,*** равного π*X.

7. Область видимости переменной

В программах на ***VBA*** все идентификаторы имеют некоторую область видимости (scope). Эта область кода, в пределах которой определена данная переменная.

Переменная, объявленная внутри функции, не может быть использована или обнаружена вне данной функции. Это означает, что можно использовать одно и то же имя переменной в нескольких пользовательских функциях. Компилятор ***VBA*** определит соответствие используемых переменных. областям видимости.

Это позволяет все создаваемые пользователем функции записывать в одном модуле.

8. Тип Variant и особенности его использования

Тип ***Variant*** удобен при записи коротких программ. Он требует при записи переменных больше ресурсов памяти и больше времени для обработки. При обработке переменных, объявленных как ***Variant,*** компилятор преобразует эту переменную к типу, соответствующему значению переменной, после чего выполняются операции, определяемые записанными в программе выражениями.

9. Расположение нескольких операторов в одной строке

Для того, чтобы уменьшить число строк в записываемой программе, между операторами следует поставить двоеточие. Например, две, приводимые ниже последовательности, эквивалентны.

```
X = X + 1
H = H + X*10
```

и

```
X = X + 1 : H = H + X*10
```