ЛЕКЦИЯ 13
ТЕМА: ФУНКЦИЯ И ЕЁ ТИПЫ

1. **Понятие подпрограммы**
2. **Подпрограмма-процедура**
3. **Подпрограмма-функция**

1. Понятие подпрограммы

Иногда в определенных местах программы приходится выполнять практически одни и те же последовательности действий с разными исходными данными. Такие последовательности действий можно оформить в виде так называемых подпрограмм (от англ. "subroutine"). Подпрограммы сокращают текст программы, существенно уменьшают время их исполнения, облегчают жизнь программистам, которые могут создавать программы модульно, т. е. собирая сложную программу из законченных кусочков более простых составляющих. Это позволяет создавать большие программы группой программистов, разрабатывать и реализовать группе школьников какие-либо глобальные проекты.

Для примера использования процедуры приведем программу вычисления следующего выражения:

Z=N!/M!*(N-M)!

Переменные N и M, а также выражение (N—M) снабжены восклицательным знаком, который в математике означает не что иное, как факториал (например, N-факториал). Факториал — это произведение натуральных чисел от 1 до N включительно. Например, 3!=1x2x3=6, а, например, 6! — это уже 1x2x3x4x5x6=720.

В выражении нам три раза придется вычислять факториал, производя при этом одни и те же действия с разными числами. Нам облегчит работу подпрограмма, которую мы разместим в конце основной программы и назовем, например, fact. В подпрограмме мы опишем, как вычислить факториал вообще, а затем из основной программы обратимся к подпрограмме три раза с различными параметрами.

2. Подпрограмма-процедура

В Visual Basic используется два вида подпрограмм: это подпрограмма-процедура и подпрограмма-функция.

В Visual Basic, как и во многих других языках программирования, большинство программ создается из блоков - процедур и функций. Весь программный код находится как бы внутри этих процедур. Если возникает необходимость в решении какой-либо задачи в любом месте программы, то вызывается процедура. В Visual Basic нельзя ввести код между процедурами. Код всегда должен находиться внутри процедуры.

Давайте разберёмся с понятиями, и определим, что будет называться процедурой, а что функцией.

Процедуры:

Процедура - это некий блок кода, который будет выполняться всякий раз при вызове этой процедуры. Каждая процедура начинается зарезервированным словом Sub и заканчивается End. Каждой процедуре нужно присвоить название, и определить список входящих и выходящих параметров. Выходной параметр процедуры – это переменная значение, которой образуется в результате работы процедуры. Синтаксис процедуры:

Sub *имя_процедуры* (список параметров)

Программный код

End Sub

Обращение к процедуре в программе осуществляется с помощью оператора Call. Оно пишется следующим образом:

Call *имя_процедуры*([*список входящих параметров*])

В Visual Basic существует и другой метод обращения к процедуре:

***Имя_процедуры*([*список входящих параметров*])**

Процедуры бывают глобальными и локальными. Локальная процедура может быть доступна только внутри заданного программного модуля, её нельзя вызвать из других модулей. Он определяется следующим образом:

```
Private Sub имя_процедуры
  программный код
   [Exit Sub]
   [здесь тоже может быть некий программный код]
End Sub
```

Всё, что заключено в квадратные скобки - является необязательным. Оператор Exit Sub позволяет досрочно выйти из процедуры. Иногда это очень удобно.

Глобальную процедуру можно вызвать из других программных модулей. Он определяется следующим образом:

```
Public Sub имя_процедуры
  программный код
   [Exit Sub]
   [здесь тоже может быть некий программный код]
End Sub
```

3. Подпрограмма-функция

Функция - это некий блок кода, который будет возвращать значение. Этим, и только этим функции отличаются от процедур. Общий синтаксис функции:

```
[Public | Private |] [Static] Function имя_функции
([список параметров]) [As type]
   [здесь некий код]
   [имяфункции = выражение]
   [Exit Function]
   [здесь тоже может быть некий код]
   [имяфункции = выражение]
End Function
```

```
    Public Function MyFunc() As Byte
  MyFunc = 234
End Function
    c = MyFunc()
```

Когда мы говорили о выражениях, мы говорили, что MyFunc - это выражение, со значением 234. Т.е. здесь, функция MyFunc возвращает значение 234 (байт). Чтобы задать это значение, необходимо присвоить имени функции выражение. В нашем случае в качестве выражения выступает число 234.

Давайте рассмотрим более практичный пример. Напишем функцию для вычисления квадрата числа. У функции будет 1 параметр типа Integer - число для возведения в квадрат. Функция будет возвращать значение квадрата параметра. Тип возвращаемого значения - Long:

```
    Public Function Square(number As Long) As Long
  Square = number * number
End Function
```

Вызвать функцию можно так:

```
    b = Square (5)
```

А можно так, используя нашу процедуру для вывода сообщения на экран:

```
    ShowMessage Square (5)
```

А можно и так:

```
    Square 5
```

В последнем случае возвращённое функций значение уходит в никуда, но сама функция благополучно выполнится. Обратите внимание на отсутствие скобок в последнем вызове. Скобки здесь не обязательны, в отличие от двух предыдущих вызовов. Там скобки обязательны. (ну это и понятно, как бы Visual Basic без скобок догадался, где параметры для ShowMessage, а где для Square).