Методы и их параметры

Последнее обновление: 30.10.2015

Методы представляют собой набор операторов, предназначенных для выполнения определенного действия. Методы условно можно разделить на два типа: процедуры и функции. Если процедуры просто выполняют определенные действия, то функции возвращают некоторое значение.

Синтаксис процедуры выглядит следующим образом: сначала мы указываем модификатор доступа, затем ключевое слово **Sub**, после имя процедуры. После имени процедуры в скобках объявляем параметры процедуры, однако метод может и не иметь параметров - в таком случае в скобках ничего не указывается. Заканчивается объявление процедуры словами **End Sub**. С одной процедурой вы уже должны быть знакомы - это процедура Main, которая и вызывается при запуске модуля:

```
Sub Main()
    'Здесь выполняемые действия
End Sub
```

или

```
Sub Method1()
    Console.WriteLine("This is a Method1")
End Sub
```

Объявление функции похоже на объявление процедуры, только вместо ключевого слова **Sub** используется слово **Function**. Второе различие состоит в том, что нам надо указать тип и возвращаемое значение из функции. Чтобы указать тип, после скобок с параметрами помещается ключевое слово **As**, после которого пишется тип, значение которого возвращает функция. Если тип не указан, то функция по умолчанию возвращает значение типа `Object`. Кроме того, в конце функции мы помещаем слово **Return** и после него указываем возвращаемое значение:

```
Function Factorial() As Integer
    Return 1
End Function
```

или

```
Function Hello() As String
    Dim s As String = "Hello World"
    Return s
End Function
```

Использование методов в программе

Чтобы вызвать метод в программе, надо указать имя метода, а после него в скобках значения для его параметров:

```
Sub Main()
    'Присваиваем переменной message строку из функции Hello()
    Dim message As String = Hello()
    Console.WriteLine(message)
    'Выполняем метод Addition, который выводит сообщение на экран
    Addition()
    Console.ReadLine()
End Sub
'Функция, возвращающая строку Hello World
Function Hello() As String
    Dim s As String = "Hello World"
    Return s
End Function
'Метод, который выводит на экран сообщение
Sub Addition()
    Console.WriteLine("2 + 2 = {0}", 2+2)
End Sub
```

Обратите внимание, что поскольку функция возвращает значение, это значение можно присвоить другой переменной.

Передача параметров

В вышеприведенном примере мы использовали процедуры и функции без параметров. Теперь посмотрим, как используются параметры. Параметры могут передаваться в методы *по значению* и *по ссылке*. Передача по значению (наиболее распространенный способ передачи параметров) происходит следующим образом:

```
Function Addition(x As Integer, y As Integer) As Integer
    Return x + y
End Function
```

Сначала мы указываем имя параметра, а потом после слова `As` указывается тип параметра (в данном случае оба параметра имеют тип **Integer**). Также передача по значению осуществляется с помощью ключевого слова **ByVal**, а предыдущее объявление функции будет эквивалентно следующему:

```
Function Addition(ByVal x As Integer, ByVal y As Integer) As Integer
    Return x + y
End Function
```

Передача параметров по ссылке происходит также, только вместо ByVal используется ключевое слово **ByRef**:

```
Sub Addition(ByRef x As Integer, ByVal y As Integer)
    x=x+y
End Sub
```

В чем же отличие передачи аргумента по ссылке от передачи по значению? При передаче аргумента по значению метод получает не саму переменную, а ее копию. При передаче аргумента по ссылке в метод передается адрес этой переменной в памяти. И если в методе изменяется значение такой аргумента, то также изменяется и значение переменной, которая передается на его место:

```
Module Module1

    Sub Main()
        'Начальные значения переменных a и b
        Dim a As Integer = 5
        Dim b As Integer = 6
        Console.WriteLine("Начальное значение переменной a  = {0}", a)
        'Передача переменных по значению
        'После выполнения этого кода a = 5, так как мы передали лишь ее копи
        AdditionVal(a, b)
        Console.WriteLine("Переменная a после передачи по значению равна = {
        'Передача переменных по ссылке
        'После выполнения этого кода a = 11, так как мы передали саму переме
        AdditionRef(a, b)
        Console.WriteLine("Переменная a после ссылке по значению равна = {0}

        Console.ReadLine()
    End Sub
    'Передаем аргументы по значению
    Sub AdditionVal(ByVal x As Integer, ByVal y As Integer)
        x = x + y
        Console.WriteLine("x + y = {0}", x)
    End Sub

    'Передаем аргументы по ссылке
    Sub AdditionRef(ByRef x As Integer, ByVal y As Integer)
        x = x + y
        Console.WriteLine("x + y = {0}", x)
    End Sub
End Module
```

В данном случае мы объявляем две переменные a и b. У нас есть два метода, которые принимают два параметра: x и y. В обоих методах значение аргумента x приравнивается сумме x и y. Затем мы подставляем на место параметров x и y переменные a и b соответственно. В первом случае переменная передается по значению, то есть передается копия этой переменной, и она не изменяется. Во втором случае мы передаем указатель на эту переменную в памяти, а поскольку аргумент x изменяется, то передаваемая на его место переменная a тоже изменяется.

Когда же надо передавать аргументы по ссылке, а когда по значению? Если необходимо изменить переменную или даже несколько переменных в одном методе, то следует передавать аргументы по ссылке. Также следует передавать по ссылке большие объекты, даже если не надо их изменять, поскольку создание их копии снижает производительность программы.

Необязательные параметры

В методах также могут использоваться и необязательные параметры. Чтобы объявить необязательный параметр, надо использовать ключевое слово **Optional**. Для такого параметра необходимо при объявлении метода объявить значение по умолчанию. Также надо учесть, что все последующие параметры после необязательного также должны быть необязательными. Например:

```
Function Add(ByVal x As Integer, ByVal y As Integer, Optional z As Integer =
    Return z + x + y + s
End Function
```

Тогда при вызове этой функции мы можем не передавать значения для двух последних параметров:

```
Sub Main()
    ' Не передаем значения для дополнительных параметров
    ' Оставшиеся параметры имеют значения по умолчанию
    Add(3, 10)
    'Передаем значение для одного дополнительного параметра
    ' Другой использует значение по умолчанию
    Add(1, 1, 1)
End Sub
```

Передача аргументов по имени

В предыдущих примерах мы передавали значения для параметров в порядке объявления этих параметров в методе. Однако можно также передавать значения для параметров по имени. При этом порядок передачи значений не зависит от порядка следования параметров. Для этого в вызове метода мы указываем имя параметра, потом двоеточие и знак равенства и затем значение для этого параметра. К примеру, передадим значения для параметров к вышеопределенной функции Add:

```
Sub Main()
    ' Необязательный параметр s использует значение по умолчанию
    Add(x:=3, z:=6, y:=8)
End Sub
```

Рекурсивные функции

Особо следует остановиться на рекурсивных функциях - это такая функция, которая вычисляется через саму себя. Посмотрим применение рекурсии в программе по вычислению факториала:

```
Function Factorial(x As Integer) As Integer
    If (x = 1) Then
        Return 1
    Else
        Return x * Factorial(x - 1)
    End If
End Function
```

Итак, у нас в данном случае задается условие, что если вводимое число не равно 1, то мы умножаем данное число на результат этой же функции, в которую в качестве параметра передается число x-1. И так, пока не дойдем того момента, когда значение параметра не будет равно единице.

Еще одним примером рекурсивной функции может служить функция для вычисления числа Фибоначчи. n-й член последовательности Фибоначчи определяется по формуле: f(n)=f(n-1) + f(n-2), причем f(0)=0, f(1)=1.

```
Function Fibbonachi(x As Integer) As Integer
    If x = 0 Then
        Return 0
    ElseIf x = 1 Then
        Return 1
    Else
        Return Fibbonachi(x - 1) + Fibbonachi(x - 2)
    End If
End Function
```

Назад **Содержание** **Вперед**

Иван Кулиберов
4 года назад edited

Код этот не работает корректно!

0 0 Ответить • Поделиться ›

Metanit Модератор ➚ Иван Кулиберов
4 года назад

что именно не работает корректно?

0 0 Ответить • Поделиться ›

Дмитрий
6 лет назад

Ай-ай-ай... Фибоначчи неправильно высчитывается.
Тело функции стоит поменять на:

```
If x < 1 Then
Return 0
ElseIf x <= 2 Then
Return 1
Else
Return Fibbonachi(x - 1) + Fibbonachi(x - 2)
End If
```

0 0 Ответить • Поделиться ›

Metanit Модератор ➚ Дмитрий
6 лет назад

все правильно высчитывается, единственное, что там условие f(0)=0

0 0 Ответить • Поделиться ›

Помощь сайту

YooMoney:
410011174743222

Qiwi:
qiwi.com/n/METANIT

Перевод на карту
Номер карты:
4048415020898850

Вконтакте | Телеграм | Twitter | Помощь сайту

Контакты для связи: metanit22@mail.ru