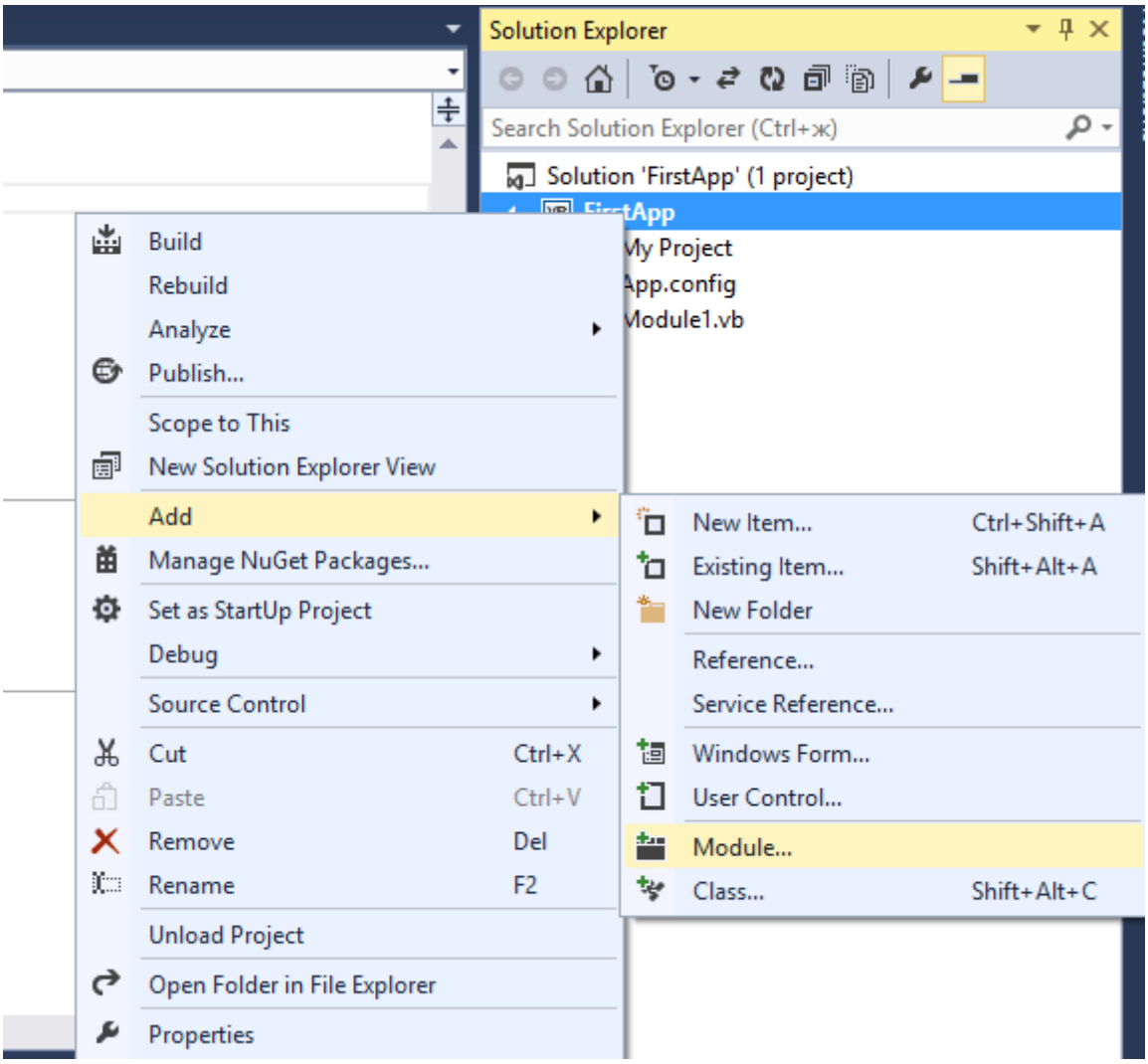
Модули

Последнее обновление: 30.10.2015

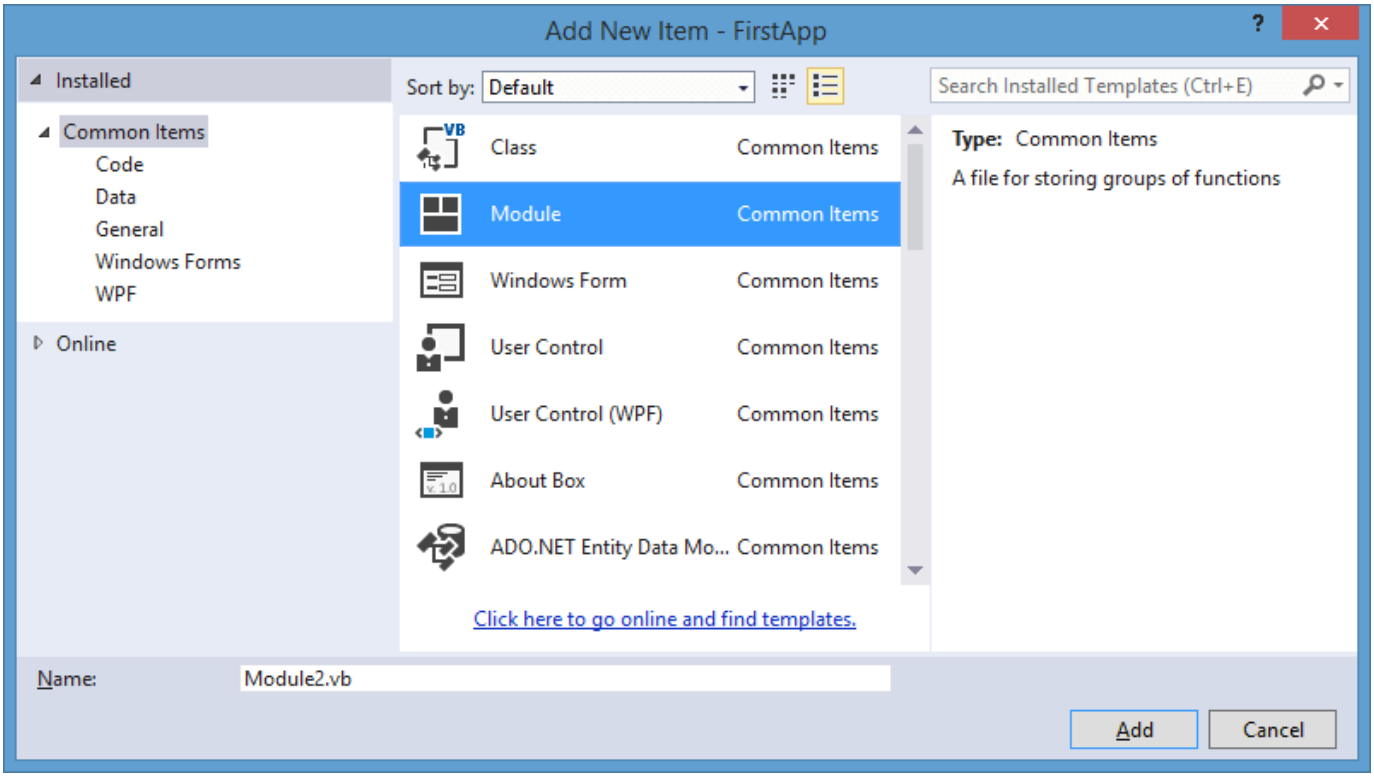
Поскольку Visual Basic.NET является полноценным объектно-ориентированным языком, для организации программного кода используются классы. Либо также могут использоваться модули. При создании нового консольного приложения Visual Studio автоматически генерирует следующий код:

```
Module Module1

    Sub Main()

    End Sub

End Module
```

В данном случае наша программа представляет модуль с именем `Module1`. Концепция модулей представляет парадигму модульного программирования, согласно которому вся программа делится на ряд модулей, которые отвечают за разные функции программы. Пока мы использовали только один модуль в программе. Теперь создадим программу из двух модулей - первый модуль будет считывать из файла некоторое значение, а другой модуль будет получать это значение и проводить с ним некоторые операции. Чтобы добавить в программу второй модуль, нажмите справа в окне **Solution Explorer** (Обозреватель решений) на название проекта правой кнопкой мыши, затем в появившемся списке выберите пункт **Add (Добавить) -> Module...(Модуль)**.



В открывшемся диалоговом окне выберите пункт **Module (Модуль)**, оставьте в качестве его имени **Module2** и нажмите кнопку **Add (Добавить)**



Таким образом, мы добавили в программу новый модуль **Module2**. В его коде ничего не определено, кроме объявления самого модуля:

```
Module Module2

End Module
```

Этот модуль будет отвечать у нас за считывание значения из файла. Чтобы считать файл, воспользуемся классом **StreamReader**, определенным в пространстве имен **System.IO**. Поэтому нам нужно импортировать данное пространство имен с помощью оператора **Imports**. Импортирование пространства имен производится в самом начале программы перед определением модуля или класса:

```
Imports System.IO
Module Module2

End Module
```

Что такое пространство имен? Пространства имен являются контейнерами для модулей, классов и других пространств имен. Одно и или несколько пространств имен и составляют приложения или библиотеки dll, построенные на платформе .NET. Мы можем и наш модуль поместить в пространство имен, которое назовем к примеру **Modules**. Это делается с помощью ключевого слова **Namespace**:

```
Imports System.IO
Namespace Modules
    Module Module2

    End Module
End Namespace
```

Теперь перейдем к самой реализации нашей программы - определим функцию, которая будет в качестве параметра принимать путь к файлу и будет возвращать считанное значение:

```
Imports System.IO
Namespace Modules
    Module Module2
        Function Read(path As String) As Integer
            'Число, которое возвращаем из функции
            Dim number As Integer
            Try
                'Поток для считывания
                Dim sr As New StreamReader(path)
                'Считываем первый символ в файле
                number = Int32.Parse(sr.ReadLine())
                'Закрываем поток
                sr.Close()
```

```
            Catch ex As Exception
                Console.WriteLine(ex.Message)
            End Try
            'Возвращаем результат
            Return number
        End Function
    End Module
End Namespace
```

Обратите внимание на конструкцию **Try ... Catch ... End Try** - она нужна нам для обработки ошибок. Мы могли бы ее не использовать, но при выполнении программы может возникнуть ошибка. Например, мы введем неверный путь к файлу, и чтобы программа не зависла, а продолжала работать, мы используем данную конструкцию. После выражения **Catch** определен код для вывода ошибки на экран: **Console.WriteLine(ex.Message)**.

Весь код нашей программы сосредоточен в трех строках между Try и Catch:

```
Dim sr As New StreamReader(path)
number = Int32.Parse(sr.ReadLine())
sr.Close()
```

В первой строке мы создаем поток для считывания файла, который мы получаем из параметра path. Чтобы создать новый объект используется ключевое слово **New**. Во второй строке мы считываем первый символ из файла. Метод **ReadLine** класса StreamReader считывает одну строку из файла, поэтому нам надо будет потом ее привести к типу Integer и полученное значение присвоить переменной number. В третьей строке мы закрываем поток методом `Close`.

Теперь перейдем к главному модулю. Он будет получать результат из модуля Module2 и вычислять факториал числа:

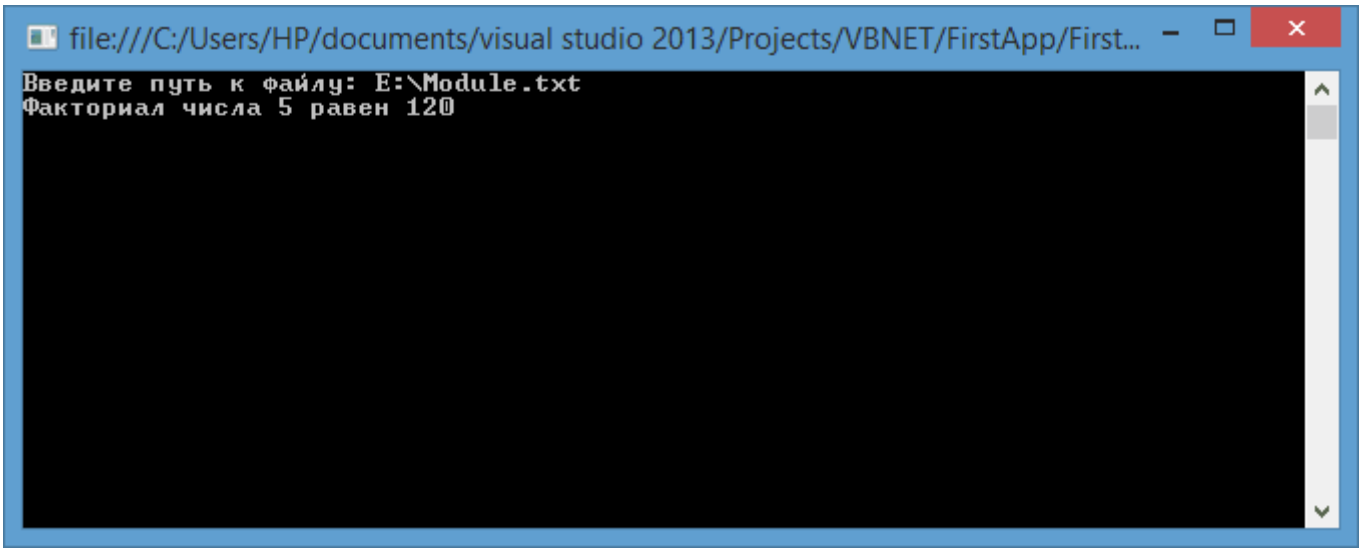
```
Module Module1

    Sub Main()


        Console.Write("Введите путь к файлу: ")
        'Вводим с клавиатуры полный путь к файлу
        Dim path As String = Console.ReadLine()
        'Получаем число из файла, используя модуль Module2
        Dim number As Integer = Modules.Module2.Read(path)
        Console.WriteLine("Факториал числа {0} равен {1}", number, Factorial
        Console.ReadLine()
    End Sub
    Function Factorial(x As Integer) As Integer
        If (x = 1) Then
            Return 1
```

```
        Else
            Return x * Factorial(x - 1)
        End If
    End Function
End Module
```

Итак, этот код должен быть вам в целом знаком: сначала мы вводим путь к файлу, передаем его в функцию `Read`, которая определена в модуле `Module2`. Так как мы для модуля `Module2` определили пространство имен, то надо указать и его, поэтому вызов метода имеет следующий вид:

Пространство_имен_модуля.Модуль.Метод_модуля

Хотя, если бы оба модуля находились в одном пространстве имен, то мы бы могли не указывать пространство имен. Итак, допустим, у нас на диске E находится файл Module.txt, в которое записано число 5. Тогда запустим программу, нажав F5, введем полный путь к файлу и получим факториал числа 5:



Назад Содержание Вперед

ТАКЖЕ НА **METANIT.COM**

Клиент на Xamarin Forms для SignalR

10 дней назад · 1 коммента…

Клиентское приложение на Xamarin Forms для SignalR в ASP.NET Core, …

Параметры строки запроса

5 месяцев назад · 1 коммен…

Параметры строки запроса query string в приложении Blazor на …

Отправка запросов на сервер. HttpClient

5 месяцев назад · 1 коммен…

Отправка запросов на сервер HttpServer с помощью класса …

Взаи кодо

5 меся

Взаим Pythor языке