Если переменные хранят некоторые значения, то методы содержат собой набор инструкций, которые выполняют определенные действия. По сути метод - это именованный блок кода, который выполняет некоторые действия.

Общее определение методов выглядит следующим образом:

|  |  |
| --- | --- |
| 1  2  3  4 | [модификаторы] тип\_возвращаемого\_значения название\_метода ([параметры])  {  // тело метода  } |

Модификаторы и параметры необязательны.

Ранее мы уже использовали как минимум один метод - **Console.WriteLine()**, который выводит информацию на консоль. Теперь рассмотрим, как мы можем создавать свои методы.

### Определение метода

Определим один метод:

|  |  |
| --- | --- |
| 1  2  3  4 | void SayHello()  {  Console.WriteLine("Hello");  } |

Здесь определен метод SayHello, который выводит некоторое сообщение. К названиям методов предъявляются в принципе те же требования, что и к названиям переменных. Однако, как правило, названия методов начинаются с большой буквы.

Перед названием метода идет возвращаемый тип данных. Здесь это тип **void**, который указыает, что фактически ничего не возвращает, он просто производит некоторые действия.

После названия метода в скобках идет перечисление параметров. Но в данном случае скобки пустые, что означает, что метод не принимает никаких параметров.

После списка параметров в круглых скобках идет блок кода, который представляет набор выполняемых методом инструкций. В данном случае блок метода SayHello содержит только одну инструкцию, которая выводит строку на консоль:

|  |  |
| --- | --- |
| 1 | Console.WriteLine("Hello"); |

Но если мы запустим данный проект, то мы не увидим никакой строки, которую должен выводить метод SayHello. Потому что после определения метод еще надо вызвать, чтобы он выполнил свою работу.

### Вызов методов

Чтобы использовать метод SayHello, нам надо его вызвать. Для вызова метода указывается его имя, после которого в скобках идут значения для его параметров (если метод принимает параметры).

|  |  |
| --- | --- |
| 1 | название\_метода (значения\_для\_параметров\_метода); |

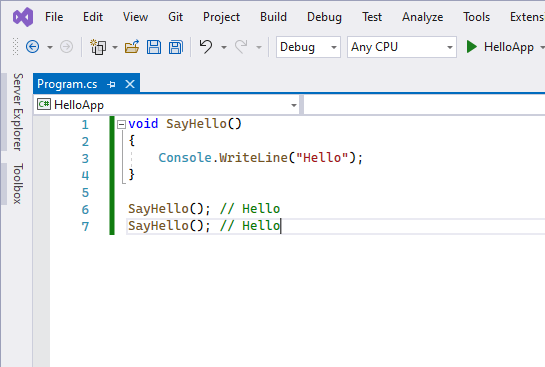
Например, вызов метода SayHello будет выглядеть следующим образом:

|  |  |
| --- | --- |
| 1 | SayHello(); |

Поскольку метод не принимает никаких параметров, то после названия метода идут пустые скобки.

Объединим определение и вызов метода:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | void SayHello()  {  Console.WriteLine("Hello");  }    SayHello(); // Hello  SayHello(); // Hello |



Консольный вывод программы:

Hello  
Hello

Преимуществом методов является то, что их можно повторно и многократно вызывать в различных частях программы. Например, в примере выше два раза вызывается метод SayHello.

При этом в данном случае нет разницы, сначала определяется метод, а потом вызывается или наоборот. Например, мы могли бы написать и так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SayHello(); // Hello  SayHello(); // Hello    void SayHello()  {  Console.WriteLine("Hello");  } |

Определим и вызовем еще несколько методов:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | void SayHelloRu()  {  Console.WriteLine("Привет");  }  void SayHelloEn()  {  Console.WriteLine("Hello");  }  void SayHelloFr()  {  Console.WriteLine("Salut");  }      string language = "en";    switch (language)  {  case "en":  SayHelloEn();  break;  case "ru":  SayHelloRu();  break;  case "fr":  SayHelloFr();  break;  } |

Здесь определены три метода SayHelloRu(), SayHelloEn() и SayHelloFr(), которые также имеют тип void, не принимают никаких параметров и также выводит некоторую строку на консоль. Условно говоря, они выводят приветствие на определенном языке.

В конструкции switch проверяется значение переменной language, которая условно хранит код языка, и в зависимости от ее значения вызывается определенный метод. Так, в данном случае на консоль будет выведено

Hello

### Сокращенная запись методов

Если метод в качестве тела определяет только одну инструкцию, то мы можем сократить определение метода. Например, допустим у нас есть метод:

|  |  |
| --- | --- |
| 1  2  3  4 | void SayHello()  {  Console.WriteLine("Hello");  } |

Мы можем его сократить следующим образом:

|  |  |
| --- | --- |
| 1 | void SayHello() => Console.WriteLine("Hello"); |

То есть после списка параметров ставится оператор **=>**, после которого идет выполняемая инструкция.