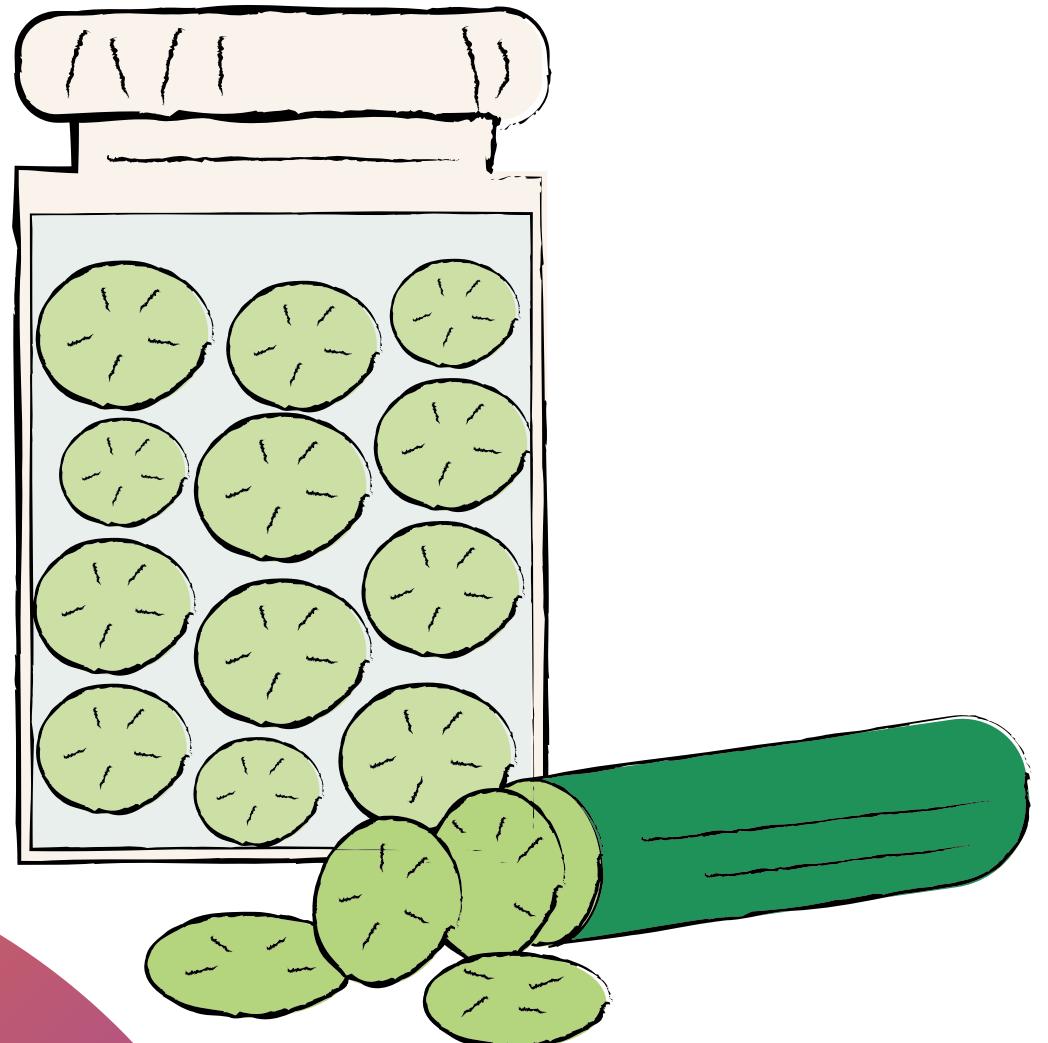




ASICODE

Курс Основи - C#

ЗАНЯТТЯ № 7 - МЕТОДИ



Визначення методу

Якщо змінні зберігають деякі значення, то методи містять набір інструкцій, які виконують певні дії. Насправді спосіб - це іменований блок коду, який виконує деякі дії.

Загальне визначення методів виглядає так:

```
тип_возвращаемого_значения название_метода ([параметры])
{
    // тело метода
}
```

Модифікатори та параметри необов'язкові.

Приклад:

```
void SayHello()
{
    Console.WriteLine("Hello");
}
```

Тут визначено метод SayHello, який виводить деяке повідомлення. До назв методів пред'являються у принципі самі вимоги, як і назв змінних. Однак, як правило, назви методів починаються з великої літери.



hello

Виклик методів

Щоб використати метод `SayHello`, нам треба його викликати. Для виклику методу вказується його ім'я, після якого у дужках йдуть значення його параметрів (якщо метод приймає параметри).

название_метода (значения_для_параметров_метода);

Наприклад, виклик методу `SayHello` виглядатиме так:

`SayHello();`

Оскільки метод не приймає жодних параметрів, після назви методу йдуть порожні дужки.



Скорочений запис методів

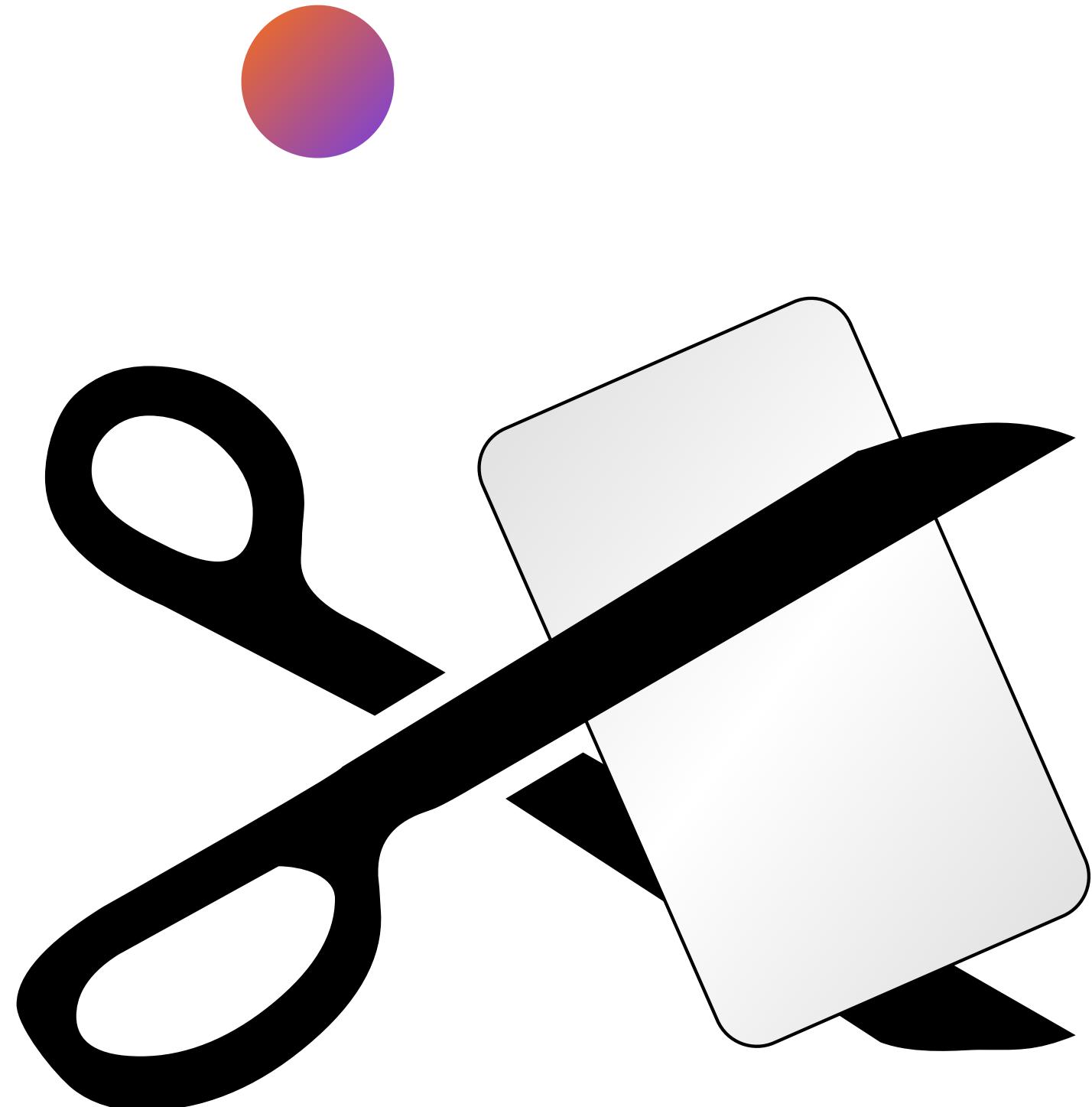
Якщо метод як тіло визначає лише одну інструкцію, ми можемо скоротити визначення методу. Наприклад, припустимо, у нас є метод:

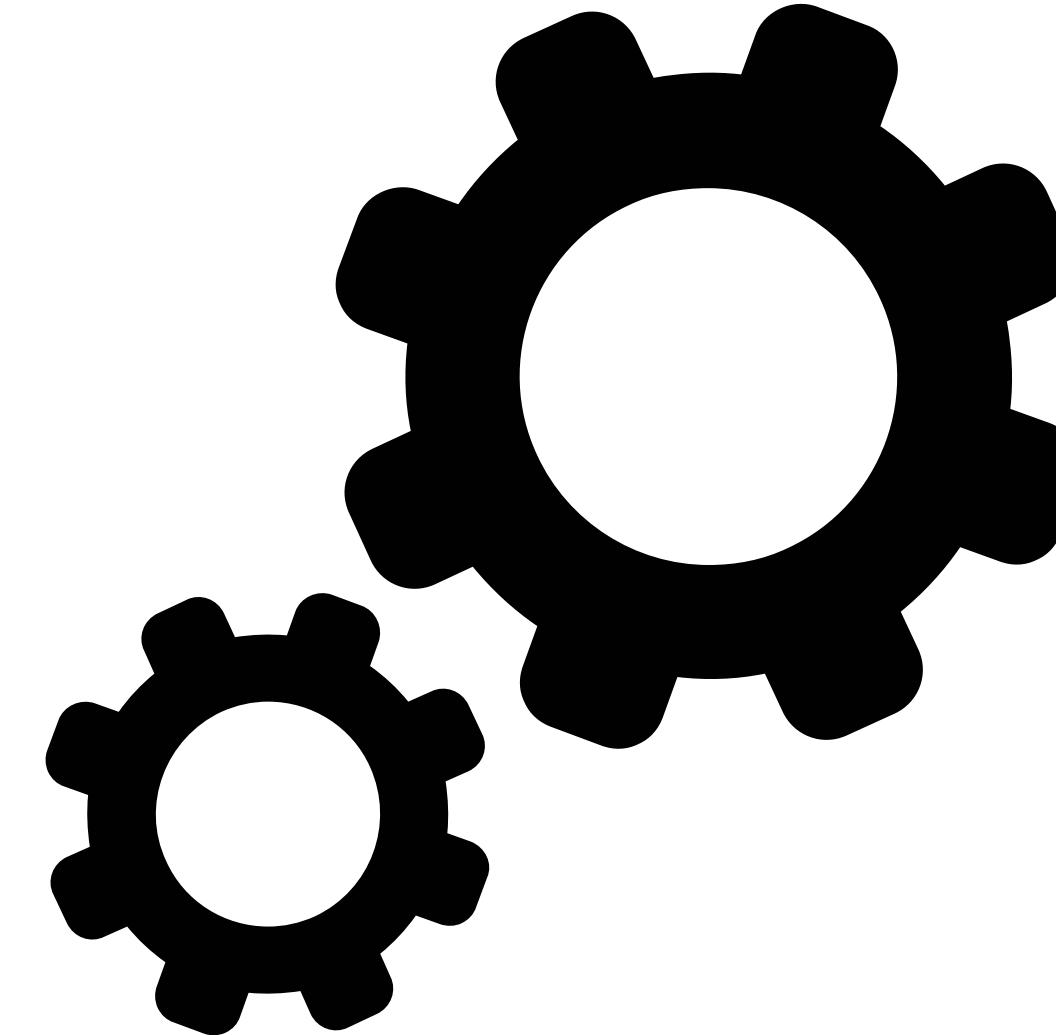
```
void SayHello()
{
    Console.WriteLine("Hello");
}
```

Ми можемо його скоротити так:

```
void SayHello() => Console.WriteLine("Hello");
```

Тобто після списку параметрів ставиться оператор `=>`, після якого йде інструкція, що виконується.





Параметри методів

Параметри дозволяють передати до методу деякі вхідні дані. Параметри визначаються через кому в дужках після назви методу у вигляді:

```
тип_метода имя_метода (тип_параметра1 параметр1, тип_параметра2 параметр2, ...)  
{  
    // действия метода  
}
```

Визначення параметра складається з двох частин: спочатку йде тип параметра, а потім його ім'я. Наприклад, визначимо метод PrintMessage, який отримує повідомлення, що виводиться ззовні:

```
void PrintMessage(string message)  
{  
    Console.WriteLine(message);  
}
```

Тут метод PrintMessage() приймає один параметр, який називається message та має тип string.

Щоб виконати метод, який має параметри, під час виклику після імені методу в дужках йому передаються значення для його параметрів, наприклад:

```
PrintMessage("Hello work");
```



Необов'язкові параметри

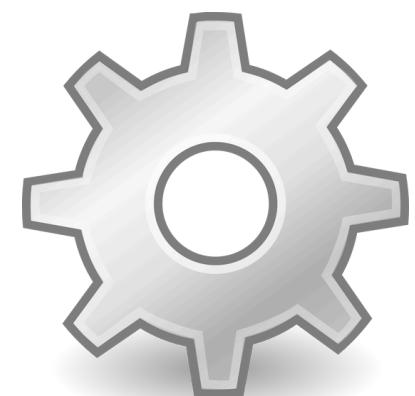
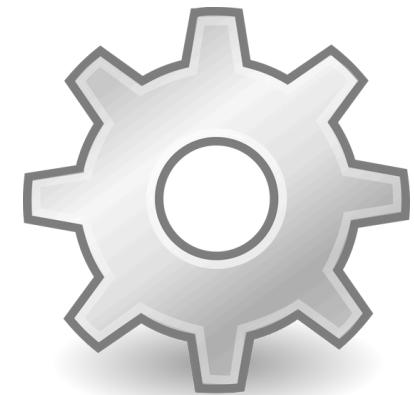
За промовчанням під час виклику методу необхідно надати значення для всіх його параметрів. Але C# дозволяє використовувати необов'язкові параметри. Для таких параметрів нам необхідно оголосити значення за промовчанням. Також слід враховувати, що після необов'язкових параметрів усі наступні параметри також мають бути необов'язковими:

```
void PrintPerson(string name, int age = 1, string company = "Undefined")
{
    Console.WriteLine($"Name: {name} Age: {age} Company: {company}");
}
```

Тут параметри `age` та `company` є необов'язковими, оскільки їм надано значення. Тому при виклику методу ми можемо не передавати їм дані:

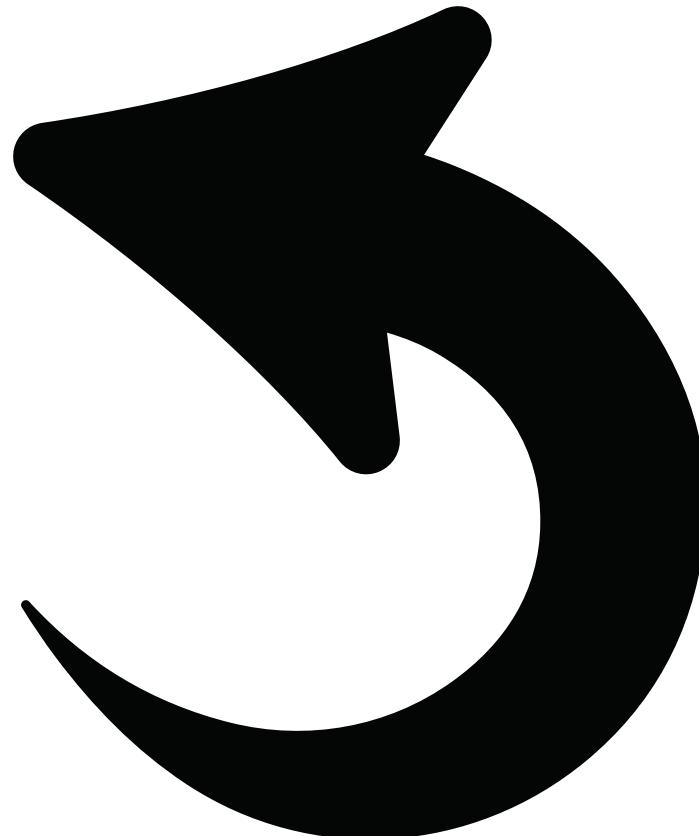
```
void PrintPerson(string name, int age = 1, string company = "Undefined")
{
    Console.WriteLine($"Name: {name} Age: {age} Company: {company}");
}
```

```
PrintPerson("Tom", 37, "Microsoft"); // Name: Tom Age: 37 Company: Microsoft
PrintPerson("Tom", 37);           // Name: Tom Age: 37 Company: Undefined
PrintPerson("Tom");
```





Возвращение значения и оператор **return**



Метод может возвращать значение, какой-либо результат. В примере выше были определены два метода, которые имели тип void. Методы с таким типом не возвращают никакого значения. Они просто выполняют некоторые действия. Но методы также могут возвращать некоторое значение. Для этого применяется оператор `return`, после которого идет возвращаемое значение:

`return` возвращаемое значение;

Например, определим метод, который возвращает значение типа string:

```
string GetMessage()
{
    return "Hello";
}
```

Метод `GetMessage` имеет тип `string`, следовательно, он должен возвратить строку. Поэтому в теле метода используется оператор `return`, после которого указана возвращаемая строка.



Передача параметрів за посиланням та значенням

Передача параметрів за значенням

Найбільш простий спосіб передачі параметрів представляє передача за значенням, по суті, це звичайний спосіб передачі параметрів:

```
void Increment(int n)
{
    n++;
    Console.WriteLine($"Число в методе Increment: {n}");
}

int number = 5;
Console.WriteLine($"Число до метода Increment: {number}");
Increment(number);
Console.WriteLine($"Число после метода Increment: {number}");

Консольний висновок:
Число до методу Increment: 5
Число в методі Increment: 6
Число після методу Increment: 5
```

При передачі аргументів параметрам за значенням параметр методу отримує саму змінну, та її копію і далі працює із цією копією незалежно від самої змінної.

Так, вище за виклик метод `Increment` отримує копію змінної `number` і збільшує значення цієї копії. Тому в самому методі `Increment` бачимо, що значення параметра `n` збільшилося на 1, але після виконання методу змінна `number` має колишнє значення - 5. Тобто змінюється копія, а сама змінна не змінюється.

Передача параметрів за посиланням та модифікатор `ref`

При надсиланні параметрів за посиланням перед параметрами використовується модифікатор `ref`:

```
void Increment(ref int n)
{
    n++;
    Console.WriteLine($"Число в методе Increment: {n}");
}
```

```
int number = 5;
Console.WriteLine($"Число до метода Increment: {number}");
Increment(ref number);
Console.WriteLine($"Число після метода Increment: {number}");

Консольний висновок:
Число до методу Increment: 5
Число в методі Increment: 6
Число після методу Increment: 6
```

При передачі значень параметрів посилання метод отримує адресу змінної в пам'яті. І, таким чином, якщо в методі змінюється значення параметра, що передається за посиланням, то також змінюється значення змінної, яка передається на його місце.

Так, у метод `Increment` передається посилання на саму змінну `number` у пам'яті. І якщо значення параметра `n` в `Increment` змінюється, то це призводить і до зміни змінної `number`, так як і параметр `n` і змінна `number` вказують на ту саму адресу в пам'яті.

Зверніть увагу, що модифікатор `ref` вказується як перед параметром під час оголошення методу, так і під час виклику методу перед аргументом, який передається параметру.



ASICODE

Дякую за увагу