C# book

Андрій Янковський

Київ 2021

C# (Сі-Шарп) – мова програмування, яка в першу чергу спрямована на створення програм під Windows. Але ви можете використовувати цю мову для створення програм, які будуть працювати на мобільних телефонах, гральних консолях, сайтах, Мак ОС, Лінукс тощо

Найкращою програмою для написання коду мовою C# є Visual Studio, тобто це програма, в якій створюють інші програми

Якщо у вас версія 2013, 2015:

Файл –> Новий –> Проект

–> Visual C# Консольна програма (Console Application)

Якщо у вас версія 2017, 2019:

–> C# Console App (.NET Framework)

Давайте допишемо два рядки в базовий код:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PershaPrograma
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Tak");
            Console.ReadLine();
        }
    }
}
```

Тепер натиснемо кнопку

І на екрані з’явиться слово Tak

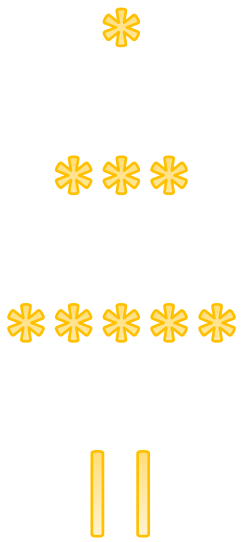
в чорному вікні. Це вікно і є консольною програмою.

Давайте виведемо на екран слово “Hello”, для цього нам потрібна команда:

```
Console.WriteLine("Hello");
```

А тепер намалюємо ялинку:

```
Console.WriteLine("    *    ");
Console.WriteLine("   ***   ");
Console.WriteLine(" ***** ");
Console.WriteLine("    ||    ");
```



Також в C# є команда, яка виводить текст, не переносячи курсор вниз:

```
Console.Write("    *     ");
Console.Write("   ***    ");
Console.Write(" ***** ");
Console.Write("    ||    ");
```

```
    *        ***      *****      ||
```

І є можливість закоментувати код, тобто, не виконувати його:

```
// Console.Write("   ***    ");
```

або так:

```
/* Console.Write("    *     ");
Console.Write("   ***    "); */
```

Для того, щоб додати два числа ми маємо їх кудись записати :)

Для зберігання цифр та текстів використовуються змінні:

```
int a = 2;
int b = 3;
int c = a + b;
Console.WriteLine(c);
```

Виведеться: 5

Тут команда int перед змінною a говорить програмі, що в змінну a можна записувати лише цифри

~~int x = "Hello";~~ неправильно!

int y = 2; правильно!

А як нам записати текст в змінну?

```
string z = "Hello";
```

А що буде, якщо додавати не цифри, а тексти "Hello" + " C#!" ?

```
string a = "Hello";
string b = "C#!";
string c = a + " " + b;
Console.WriteLine(c);
```

Виведеться: **Hello C#!**

Ага!

Якщо додавати тексти - то вони склеюються в один рядок.

А якщо додавати цифри - то вони додаються, як в математиці.

А як в змінну записати щось таке:

$$\frac{2+4}{2}$$

Дуже просто:

~~int a = 2 + 4/2;~~ неправильно! (a = 4)

int a = (2 + 4)/2; правильно! (a = 3)

А значення змінної можна потім змінювати на інше?

Звичайно:

int a = 5; зараз a = 5

a = 12; тепер a = 12

a = a + 2; тепер a = 14

Так-так, a = a + 2 комп’ютер прекрасно розуміє і збільшує значення змінної a на 2

Як ввести слово з клавіатури?

```
string s = Console.ReadLine();
```

А для того, щоб ввести цифру, потрібна більш довга команда:

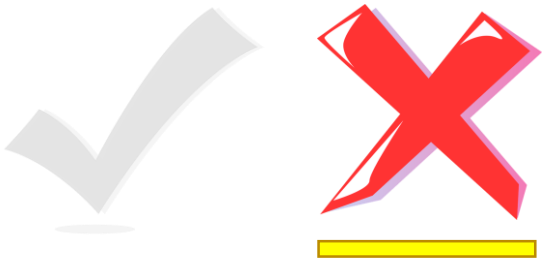
```
int a = Convert.ToInt32(Console.ReadLine());
int b = Convert.ToInt32(Console.ReadLine());
int c = a + b;
Console.WriteLine(c);
```

Ого! Тепер ми можемо додавати (+), віднімати (-), множити (*) і ділити (/) будь-які числа, що вводяться з клавіатури.

Як перевірити, чи a менше 50?

```
if (a < 50) {
  Console.WriteLine("U mene menshe 50 hrn");
}
```

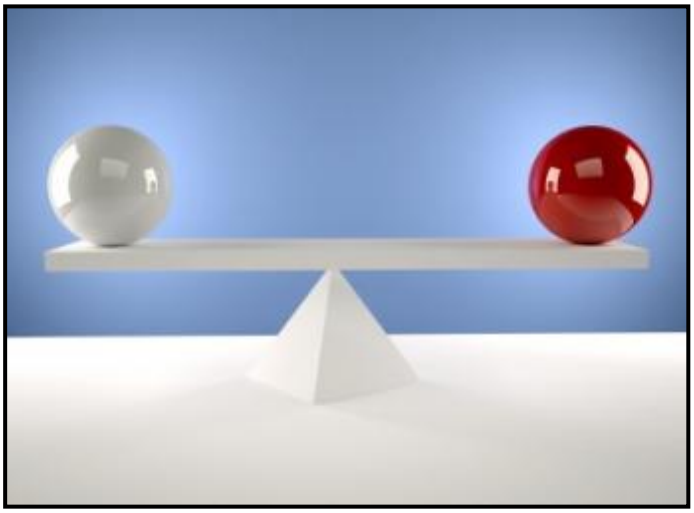
Якщо умова правильна (a < 50), то виведеться U mene menshe 50 hrn, а якщо (a >= 50) (більше або дорівнює) – то нічого не виведеться, б-е-е-е



Тут ми можемо додати ще одну команду (а в іншому випадку)

```
if (a < 50) {
    Console.WriteLine("Ya bidnyi");
}
else {
    Console.WriteLine("Ya bahatyi");
}
```

Якщо умова правильна (a < 50) , то виведеться на екран Ya bidnyi, а якщо ні - Ya bahatyi



А тепер давайте перевіримо, чи одне число дорівнює іншому

Може так:

~~if(a = 50){ }~~ неправильно!

бо = це команда для запису в змінну a числа 50

А якщо так:

if(a == 50){ } правильно!

бо == це команда перевірки, чи в змінній a теж знаходиться число 50

А тепер давайте виведемо на екран комп’ютера 10 раз слово “Hello”

```
for (int i = 0; i < 10; i++){
    Console.WriteLine("Hello");
}
```

Ця команда запускає цикл for.

Перший раз цикл запускається зі значенням i = 0 і виводить “Hello”

На другий раз спрацьовує команда i++ (збільшити i на 1)

і цикл виводить “Hello” (i = 1)

і знову виводить “Hello” (i = 2)

і знову виводить “Hello” (i = 3)

і так поки i не стане 10, бо тоді умова (i < 10) не спрацює і цикл закінчить свою роботу.

А що буде, якщо 5:2 ?

```
int a = 5 / 2;
```

Тоді a = 2;

Річ у тім, що зміна типу int не може зберігати дробову частину числа. Тип int її просто відкидає:

```
int a = 2.9;      // a = 2
int a = 2.1;      // a = 2
```

Для цифр, які можуть мати дробову частину, є вдвічі більший (подвійний) тип даних:

```
double a = 2.5;        // a = 2.5
```

Давайте спробуємо тепер

double a = 5 / 2;

І знову a = 2; // ??????

Річ у тім, що коли ми ділили int / int то ще справа від знаку

=

програма вже помістила результат у більший з двох типів даних, тобто в int, а він відкинув 0.5
Отже ми маємо сказати програмі, що чисельник чи знаменник має тип double:

double a = (double)5 / 2;

Тепер a = 2.5; // :)

Щоб згенерувати випадкове число, ми маємо спочатку об’явити спеціальну змінну типу Random

```
Random rand = new Random();
```

Тепер ми можемо генерувати випадкові числа, просто вказавши потрібний діапазон:

```
int a = rand.Next(1, 100);   // (1 – 99)
Console.WriteLine(a);        // 45
```

А тепер калькулятор!

1. Вводимо два числа a, b
2. Вводимо знак (+,-,*,/)
3. Об'являємо змінну result = 0
4. Додаємо 4 перевірки if
5. І виводимо результат

```
Console.WriteLine("Vvedit a: ");
double a =
    Convert.ToDouble(Console.ReadLine());
Console.WriteLine("Vvedit b: ");
double b =
    Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Vvedit znak: ");
string znak = Console.ReadLine();

double result = 0;

if (znak == "+") { result = a + b; }
if (znak == "-" ) { result = a - b; }
if (znak == "*") { result = a * b; }
if (znak == "/" ) { result = a / b; }

Console.WriteLine("Rezultat: " + result);
```

Цикли

З їх допомогою ми можемо виконати одну чи кілька команд багато разів.

Наприклад, давайте виведемо слово “Ya mozhu” п'ять разів.

```
for ( int i = 0; i < 5; i++ ){
    Console.WriteLine("Ya mozhu.");
}
```

Виведеться:

Ya mozhu.

Ya mozhu.

Ya mozhu.

Ya mozhu.

Ya mozhu.

Цей цикл називається for (фор)

В середині циклу об’являється змінна i, яка буде доступна тільки всередині фігурних дужок {} циклу for

Цю змінну можна вивести на екран:

```
for ( int i = 0; i < 5; i++ ){
    Console.WriteLine(i);
}
```

Виведеться:

0

1

2

3

4

Всередині циклу ми об’явили змінну i та присвоїли їй значення 0.
Саме тому при першому запуску цикл вивів на екран значення 0, потім спрацювала команда i++
(збільшити зміну i на 1)
і на другому запуску цикл вже вивів 1, на третьому - 2, на четвертому - 3, на п’ятому - 4, а коли змінна i стала дорівнювати 5, умова i < 5 (5 < 5) перестала виконуватися і цикл закінчив роботу.

Далі будуть виконуватися команди, що стоять після циклу.

А є команда i-- ?

Звичайно!

Вона зменшує зміну i на 1

А якщо ми хочемо вивести на екран такі цифри: 1, 3, 5, 7, 9

```
for ( int i = 1; i < 10; i = i + 2 ){
    Console.Write(i + ", ");
}
```

Виведеться:

1, 3, 5, 7, 9,

Цикл for складається з трьох частин, які розділяються між собою знаком ;

Початкове значення: i = 1

Умова: i < 10

доки умова виконується - цикл працює

Крок: як буде змінюватися змінна i при наступному запуску циклу

А є інші цикли, не for?

Звісно є!

```
int a = 0;
while ( a < 5 )
{
    a++;
    Console.Write(a + " ");
}
```

1 2 3 4 5

Цикл while більш компактний, якщо у вас вже об’явлена зміна (над циклом), яка буде бігати в циклі, а ще цю змінну можна використовувати і після того, як цикл завершить роботу.

Ще є цикл do-while, який використовується дуже рідко, його особливість в тому, що він спочатку робить команди в фігурних дужках {}, а вже потім перевіряє умову

```
int a = 10;
do
{
    a++;
    Console.Write(a);
}
while ( a < 5 );
```

10

Тобто, якщо вам треба виконати якість команди мінімум один раз, незалежно від значення змінної a, тоді цикл do-while - те що вам потрібно!

А бувають безкінечні цикли?

Так звісно!

Якщо замість умови написати:

true - правда

```
while ( true )
{
    Console.Write("0 ");
}
```

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...

А як тоді вийти з безкінечного циклу?

Для цього є команда break;

```
int a = 0;
while ( true )
{
    Console.Write("0 ");
    a++;
    if ( a > 5 ){  break;  }
}
```

0 0 0 0 0

А ми можемо в середині циклу сказати, а далі нічого не робити і перейти до наступного запуску.

Звісно! Для цього є команда continue

```
int a = 0;
while ( a < 7 )
{
    Console.Write("0");
    a++;
    if ( a > 3 ){  continue;  }
    Console.Write("1 ");
}
```

01 01 01 0000

Також в C# є спеціальний цикл,
який сам знає коли йому зупинитися

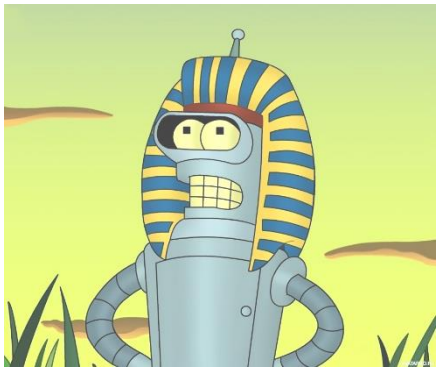
```
foreach ( int element in mas3 )
{
    Console.WriteLine(element);
}
```

Масив

Це набір фіксованої кількості однотипних елементів, що мають порядковий номер і спільне ім'я

Ось так виглядає пустий масив на 5 елементів:

```
int[] mas = new int[5];
```



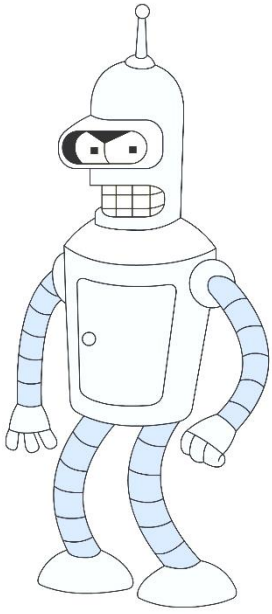
int[] – каже нам, що це масив цілих чисел, mas – його назва,
а команда = new int[5] - виділяє пам'ять під 5 елементів масиву.

А як записати якесь значення в перший елемент масиву?

mas[0] = 0;

А в інші елементи?

mas[1] = 7;
mas[2] = 5;
mas[3] = 3;
mas[4] = 1;



А в п’яту комірку як записати?

А у нас вже всі п’ять комірок зайняті 0, 1, 2, 3, 4, а 5-ї немає. Річ у тім, що в розмірі масива ми вказуємо кількість елементів, але їх нумерація іде не з 1, а з 0 – саме тому останній елемент масиву на одиничку менший за його розмір!

Виведемо масив mas на екран:

```
Console.Write(mas[0] + ", ");
Console.Write(mas[1] + ", ");
Console.Write(mas[2] + ", ");
Console.Write(mas[3] + ", ");
Console.Write(mas[4]);
```

0, 7, 5, 3, 1

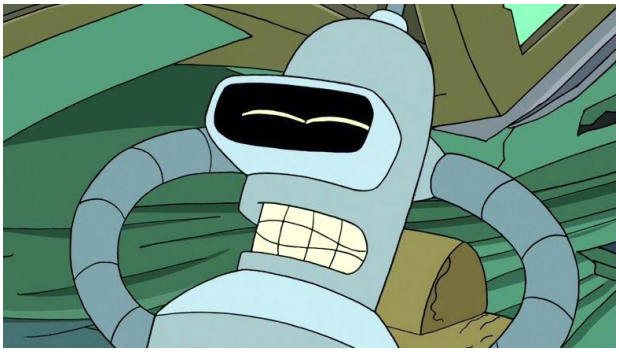
А є швидший спосіб для виводу?

Звичайно!

```
for ( int i = 0; i < 5; i++ ){
    Console.Write(mas[i] + ", ");
}
```

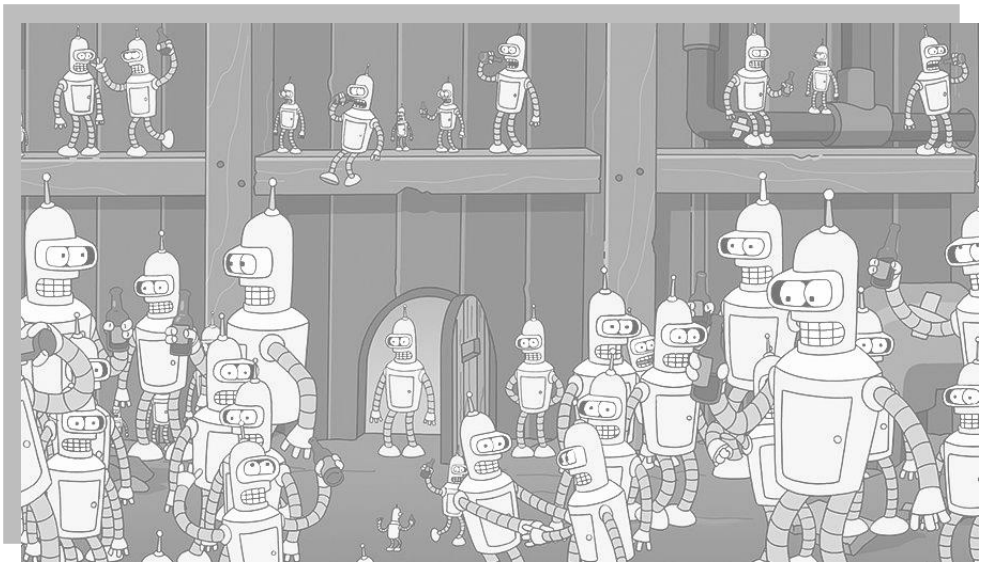
Змінна i буде змінюватися від 0 (включно) до 5 (не включно), тобто 0, 1, 2, 3, 4 і так ми зможемо кожен раз брати наступний елемент масиву та виводити його на екран:

0, 7, 5, 3, 1,



Так, ми ще можемо записати 6 цифр в шість змінних:

```
int a1 = 1;      int a2 = 3;
int a3 = 5;      int a4 = 7;
int a5 = 9;      int a6 = 11;
```



Але навіщо писати так багато, якщо є масив і ось така можливість:

```
int[] mas = { 1, 3, 5, 7, 9, 11 };
```

Не можна в один масив записувати змінні: і типу int, і типу double, і типу string – ви маєте обрати щось одне!

string[] array = { "A1", "B1", "C1" };

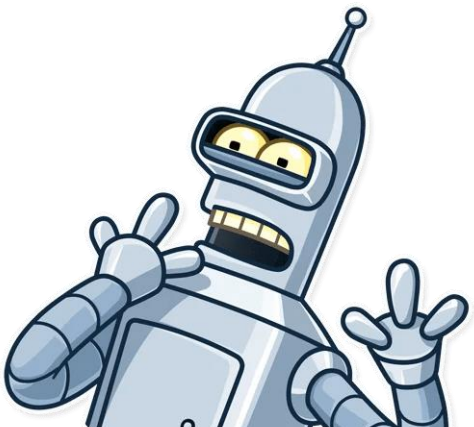
string[] array = ~~{ 1, 3.14, "C1" };~~

int[] array = ~~{ 1, 3.14, "C1" };~~

int[] array = { 6, 3, 9 };

А яка найбільша проблема при роботі з масивами?

Те, що люди часто в циклі виходять за межі масиву і програма припиняє роботу.



А можна заповнити масив автоматично?

Та без проблем!

```
for ( int i = 0; i < 6; i++ ){
    mas[i] = i;
}
```

0, 1, 2, 3, 4, 5

```
for ( int i = 0; i < 6; i++ ){
    mas[i] = i + 1;
}
```

1, 2, 3, 4, 5, 6

```
for ( int i = 6; i > 0; i-- ){
    mas[i] = i;
}
```

6, 5, 4, 3, 2, 1

```
for ( int i = 0; i < 6; i++ ){
    mas[i] = (i + 1) * (i + 1);
}
```

1, 4, 9, 16, 25, 36

```
for ( int i = 10; i <= 60; i = i + 10 ){
    mas[i] = i;
}
```

10, 20, 30, 40, 50, 60

А ми можемо заповнити масив випадковими числами?

```
Random r = new Random();

for ( int i = 0; i < 5; i++ ){
    mas[i] = r.Next(1, 10);
}
```

Так ми заповнимо масив випадковими числами від 1 до 9.

Число 1 – може зустрічатися, а 10 – ні, бо генератор генерує числа на одиницю менші від другого числа в команді r.Next(1, 10);

А давайте додамо перший елемент першого масиву з першим елементом другого масиву, а другий з другим… і запишимо результат в третій масив

```
int[] mas1 = {1, 2, 3, 4, 5};
int[] mas2 = {1, 1, 1, 1, 1};
int[] mas3 = new int[5];
```

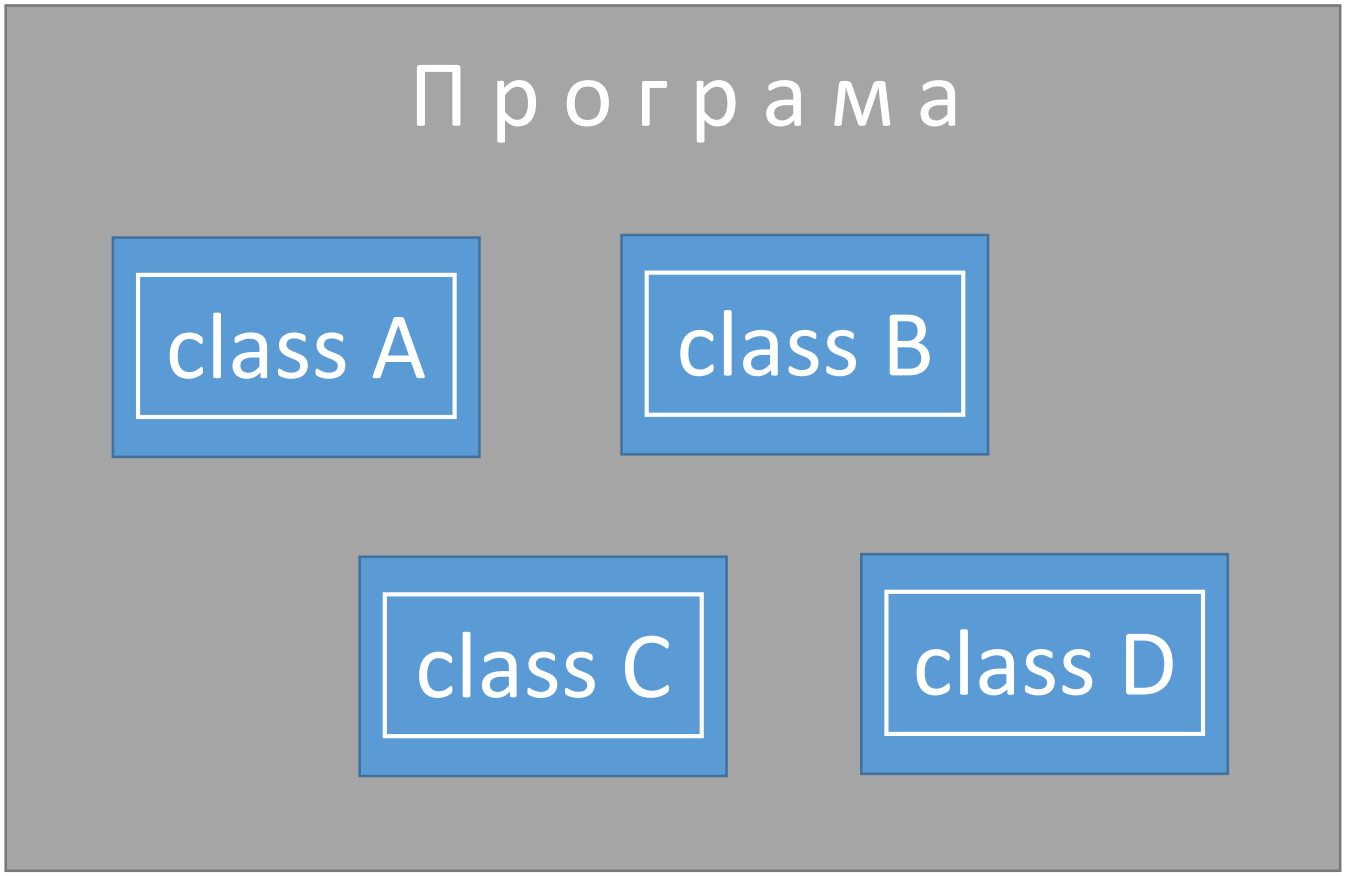
Аби створити пустий масив mas3 ми пишемо new (виділити пам’ять) int (типу int) і кількість комірок [5]

```
for ( int i = 0; i < 5; i++ ){
   mas3[i] = mas1[i] + mas2[i];
}
```

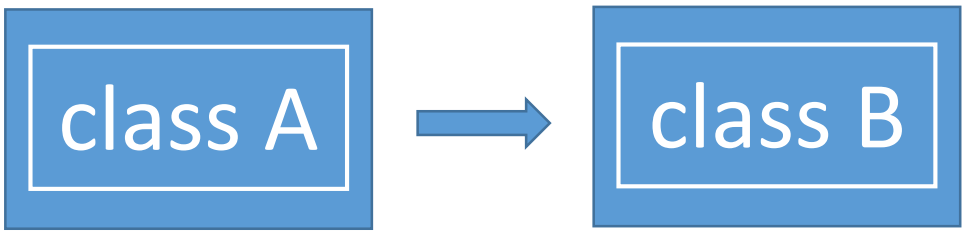
А тепер давайте спробуємо об’єднати усі значення двох масивів в третьому

```
int[] mas1 = { 1, 2, 3, 4, 5 };
int[] mas2 = { 1, 1, 1, 1, 1 };
int[] mas3 = new int[10];
int p = 0;
for ( int i = 0; i < 5; i++ ){
    mas3[p] = mas1[i];
    p++;
}
for ( int i = 0; i < 5; i++ ){
    mas3[p] = mas2[i];
    p++;
}
```

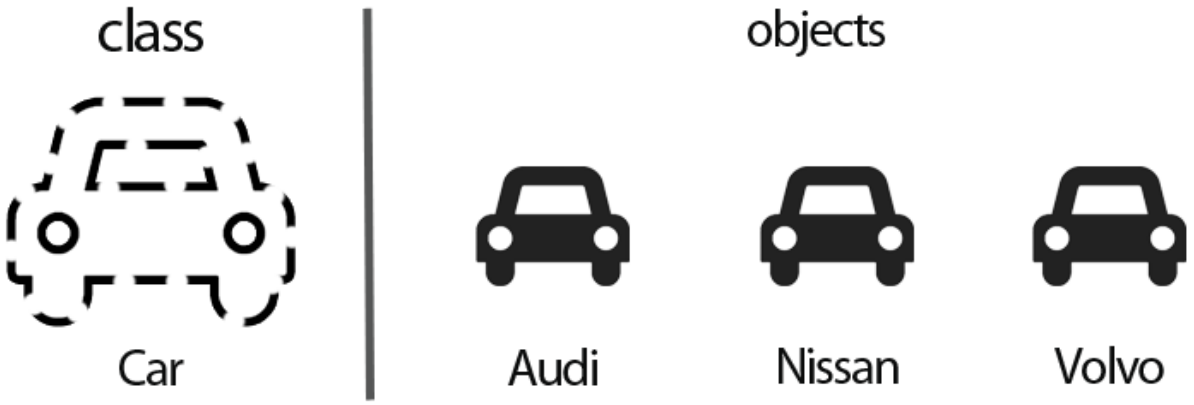
Класи розбивають велику програму на окремі модулі



Данні можна передавати з одного модуля в інший:



Клас Car – це лише опис того, що є всередині певного модуля



Аби використати клас – треба створити його змінну (об’єкт) класу:

```
Car audi = new Car();
Car nissan = new Car();
Car volvo = new Car();
```

Всередині класу можуть бути дані

```
int x;
int y;
```

і методи (функції)

```
int Add(int a, int b){
    int rezultat = a + b;
    return rezultat;
}

int Minus(int a, int b){
    int rezultat = a - b;
    return rezultat;
}
```

Данні – це змінні:

```
int a;
double b;
string c;
```

Методи – це назви блоків коду {...}. Всередині методу може і не бути команд, а може бути багато команд.

Ось пустий метод:

```
void Empty(){
    // немає команд
}
```

void – нічого не повертає в точку виклику метода.

Ось метод Message, який виводить на екран слово Hello:

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

Ось метод Add, що додає два числа і повертає результат (число типу int):

```
int Add(int a, int b){
    int rezultat = a + b;
    return rezultat;
}
```

А тепер давайте візьмемо якийсь системний клас, скажімо генератор випадкових чисел Random.
Щоб задіяти цей клас, треба спочатку об’явити змінну класу:

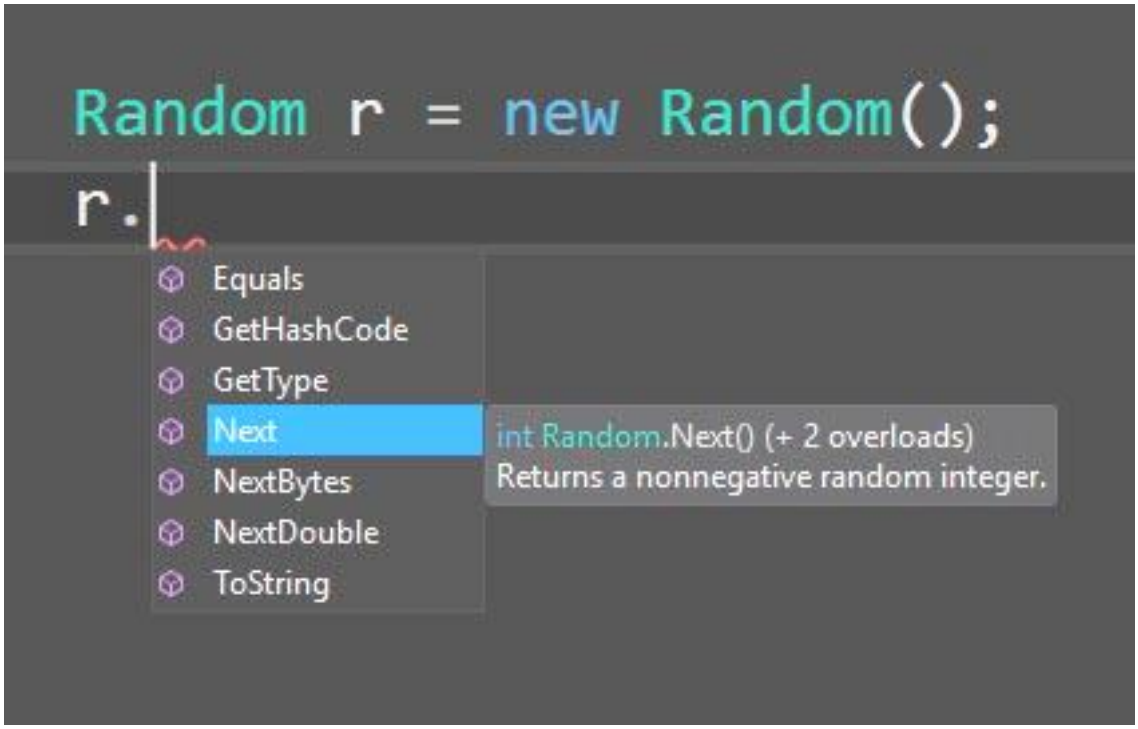
```
Random r = new Random();
```

Для того, щоб відрізнити змінну класу від звичайних змінних використовують слово об’єкт. Отже, ми щойно об’явили об’єкт r класу Random

Для того, аби об’єкт був робочім, ми виділити пам’ять під всі змінні, які в ньому знаходяться – це зробила команда new, а після new ми знову вказали назву класу але вже з ().

Як же ж тепер визивати методи класу?

Дуже просто, ми ставимо крапку після об’єкту і обираємо потрібний метод:



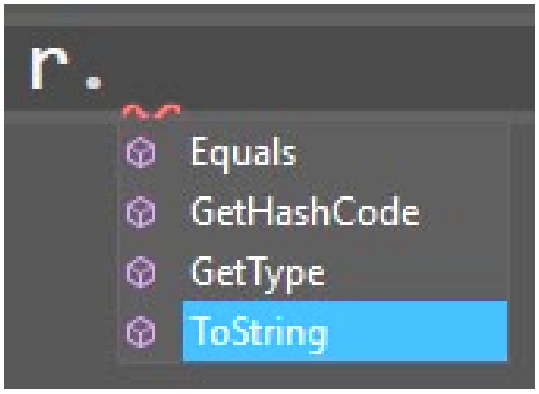
Системний клас Random підказує нам, що цей метод поверне додатне випадкове число

```
int a = r.Next(0, 10);
// 0 - 9
```

Давайте ще раз, метод Next знаходиться всередині класу Random; він приймає два числа і повертає випадкове число у відповідному діапазоні 0 – 9, тобто від першого переданого числа до другого переданого числа мінус один.

Також клас Random має метод NextDouble() – що генерує випадкове число від 0 до 1 з дробовою частиною:

```
double b = r.NextDouble();
// 0,545732851394328
```



А ці 4 стандартні методи Visual Studio додає усім об'єктам

Час настав!

Давайте створимо свій клас!

Клас Calculator з двома методами!

```
class Calculator{
  public int Add(int a, int b){
    int rezultat = a + b;
    return rezultat;
  }
  public int Minus(int a, int b){
    int rezultat = a - b;
    return rezultat;
  }
}
```

Напис public int перед назвами методів Add і Minus означає, що цей метод буде доступним для нас поза межами класу і що він повертає ціле число (int).

А як визивати методи Add і Minus?

Спочатку ми маємо створити об’єкт класу:

```
Calculator calc = new Calculator();
```

От тепер ми можемо визвати метод Add і передати туди два числа:

```
calc.Add(2, 3);
```

Але на екран нічого не виводиться?

Річ у тім, що метод Add повернув ціле число (int), але ми його ніде не використали і воно просто загубилося

А якщо так:

```
int z = calc.Add(3, 2);
Console.WriteLine("z = " + z);
```

Ось тепер на екран виведеться:

```
z = 5
```

Аналогічно:

```
int z = calc.Minus(3, 2);
Console.WriteLine("z = " + z);
```

```
z = 1
```

Ура!!! Ми написали свій перший клас!

А в класі можуть бути лише данні?

(без методів)

Звісно!

```
class Date
{
    public int day;
    public int month;
    public int year;
}
```

А як в цей клас щось записати?

Як завжди, ми маємо спочатку об’явити об’єкт класу:

```
Date d1 = new Date();
```

Надати значення змінним об'єкту:

```
d1.day = 1;
d1.month = 1;
d1.year = 2075;
```

А як вивести ці данні класу на екран?

```
Console.WriteLine(d1.day + "." +
            d1.month + "." + d1.year);
            // 1.1.2075
```

Цікаво, а є можливість зробити вивід даних через метод класу?

Щоб кожен раз оце не писати таку довгу команду.

Ми можемо створити метод Show(), який буде виводити усі дані об'єкту класу Date на екран.

Тобто, спочатку ми передамо данні в клас Date, а потім запустимо метод Show().

Класи легко розширюються, ми можемо просто додавати нові змінні і методи в них.

```
class Date{
    public int day;
    public int month;
    public int year;
    public void Show(){
      Console.WriteLine(day + "." +
                        month + "." + year);
  }
}
```

Клас сам по собі – це лише опис того, що в ньому є. Це як сліди на піску, комп'ютер ще не виділив пам'ять для змінних класу. Отже, ми маємо спочатку створити об'єкт класу

```
Date d1 = new Date();
```

Надати змінним об'єкту d1 початкові значення

```
d1.day = 1;
d1.month = 1;
d1.year = 2075;
```

і тепер визвати метод Show();

```
d1.Show();
// 1.1.2075
```

А як може виглядати клас побільше?

```
class Person
{
    public string name;
    public int age;
    public double height;
    public double weight;

    public void Walk() { }
    public void Talk() { }
    public void Eat() { }
    public void Sleep() { }
}
```

доступні поза класом дані

і методи

А ще, ми можемо створити масив не тільки з стандартних типів даних int, double, string, а й типу Person:

```
Person[] mas = new Person[3];
```

Наразі ми виділили пам'ять лише під сам масив mas, але ще не виділив пам'ять під кожен окремий об'єкт масиву:

```
mas[0] = new Person();
mas[1] = new Person();
mas[2] = new Person();
```

От тепер ми можемо заповнювати масив даними!

```
mas[0].name = "Dmytro";
mas[1].name = "Taras";
mas[2].name = "Olena";
```

Ще ми могли б додати mas[i].age mas[i].height і mas[i].weight і мати можливість вивести їх на екран, але ми просто покажемо, як це зробити на прикладі mas[i].name:

```
for ( int i = 0; i < mas.Length; i++ ){
    Console.Write(mas[i].name + ", ");
}

// Dmytro, Taras, Olena
```

Так можна вивести і 10.000 імен

Все ок, але залишається ще два питання:

1. Як класи обмінюються повідомленнями?

Визивають публічні методи один одного!

2. Якщо я не пишу перед змінною слово public, то як я це можу використати?

Це означає, що ця змінна має модифікатор доступу private (його можна не писати), тобто інші класи не мають доступу до цієї змінної! Тепер інші класи мають визвати якусь публічну функцію в нашому класі, яка до речі, може перевірити, чи у нас достатньо грошей, і тільки після цього змінити значення приватної (захищеної) змінної.

```
class Customer {
    double money;    // EUR (private)

    public void Pay(double a) {
        if (money >= a) {
            Console.WriteLine("Successful
                    payment: " + a + " EUR");
            money = money - a;
        }
        else {
            Console.WriteLine("Error");
        }
    }
    public void AddMoney(double a) {
        money = money + a;
    }
}
```

```
class Metro {
    public double price = 0.5;  // EUR
}



static void Main(string[] args) {
    Customer customer1 = new Customer();
    customer1.AddMoney(1.5);   // EUR

    Metro metro = new Metro();

    for (int i = 0; i < 5; i++)
    {
        customer1.Pay(metro.price);
    }
}
```

Після запуску програми, цикл for запуститься 5 разів, ми 5 разів спробуємо проїхатися в метро, і об'єкт customer1 класу Customer виведе нам таку інформацію:

```
Successful payment 0,5 EUR
Successful payment 0,5 EUR
Successful payment 0,5 EUR
No money
No money
```

Після перших трьох поїздок в нас закінчилися гроші 1.5 EUR і в метро нас перестали пускати!

Також один клас може наслідувати усе, що є в іншому класі

```
class BaseMetro {
    public double price;
    public void Show() { … }
}

class Metro : BaseMetro {
}
```

Тепер в класі Metro теж є змінна price та метод Show()

А можна, щось визвати з класу,
а не його об’єкту?

Так, якщо змінна або метод мають ключове слово static

```
class Metro {
    public static int life = 9;
}


Console.WriteLine(Metro.life);
// 9
```

Тепер ви вільний митець!

Ви можете знайти наступні завдання на сайтах для фриланцерів