## **Типи даних та змінні (Build-in types)**

В C# існує ділення всіх типів на 2 типи:

**·Value Types (значимі типи)** - *не можуть мати значення null:*

* **bool** - це булеве значення, що може бути як істинним (true), так і хибним (false). Вираз типу bool може бути керуючим умовним виразом в операторах ***if, do, while і for*,** а також в умовному операторі ?:. ***За замовчуванням тип bool має значення false.***
* **byte** - 8-розрядне ціле число без знаку, від 0 до 255
* **sbyte** - 8-розрядне ціле число зі знаком, від -128 до 127
* **char** - зберігає одиничний символ у кодуванні Unicode UTF-16. За замовчуванням тип char має значення \0, тобто U+0000, зберігає окремі символи, такі як 'a' або 'B'. Значення символів оточені одинарними лапками
* **float** - 32-бітний тип з плаваючою комою одинарної точності, від -3.4\*1038 до 3.4\*1038 (тіпа так 1,99 або -1,99)
* **double** - 64-бітний тип з плаваючою комою з десятковими дробами і подвійної точності, від ±5.0\*10-324 до ±1.7\*10308 (тіпа так 19,99 або -19,99)
* **decimal** - 128-бітний десятковий тип для фінансових і грошових розрахунків і може зберігати 28 знаків після коми.
* **int** - Знакове 32-бітне ціле число, від -2147483648 до 2147483647
* **uint** - Беззнакове 32-бітне ціле число, від 0 до 4,294,967,295
* **nint** - Знакове 32-бітне або 64-бітне ціле число.
* **nuint** - Беззнакове 32-бітне або 64-бітне ціле число.
* **long** - Знакове 64-бітне ціле число, від –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
* **ulong** - Беззнакове 64-бітне ціле число, від 0 до 18 446 744 073 709 551 615
* **short** - Знакове 16-бітне число, від -32768 до 32767
* **ushort** - Беззнакове 16-бітне число, від 0 до 65535
* **Var** - ключове слово, використовується для оголошення змінної неявного типу, яке визначає тип змінної на основі початкового значення.

**Syntax:** var variable\_name = value;

**Example:**

Input: a = 637

b = -9721087085262

Output: value of a 637, type System.Int32

value of b -9721087085262, type System.Int64

**·Reference type (ссилочні типи)** - *можуть мати значення null:*

* **string** - зберігає текст, наприклад "Hello World". Рядкові значення взяті в подвійні лапки.
* **arrays** - Масиви (**arrays**) використовуються для зберігання декількох значень в одній змінній, замість того, щоб оголошувати окремі змінні для кожного значення.
* **class** - основний тип даних в мові C#. Клас являє собою конструкцію, яка об’єднує поля, властивості та методи. Клас є визначенням для створення об’єктів або екземплярів класу. Якщо коротко, то класс, це місце, де записуються дані. Спочатку йде модифікатор доступу, а після ключового слова class, ім’я класу, в фігурних дужках знаходиться тіло класу, яке може містити поля, властивості та методи – **члени класу**. Клас може містити в собі наступні члени:

1. **Конструктори;**
2. **Константи;**
3. **Поля;**
4. **Методи;**
5. **Властивості;**
6. **Оператори;**
7. **Вкладені типи даних;**
8. **Деструктори.**

**Модифікатор доступу** – визначає звідки можна звертатись до класу чи його членів.

В мові C# реалізовані наступні рівни доступу:

1. **public** – максимально доступний рівень, не накладає жодних обмежень;
2. **protected** – доступ дозволено з поточного класу чи його спадкоємців;
3. **internal** – доступ обмежено поточною збіркою(в межах програми, бібліотеки);
4. **protected internal** – комбінація двох попередніх модифікаторів;
5. **private** – доступ дозволено тільки в межах даного класу;
6. **private protected** – доступ дозволено в поточному класі та класах нащадках, які містяться в цій збірці.

**Object** - об'єкт створюється з класу. Наприклад, ми створили клас з назвою “Car”, тож тепер можемо використовувати його для створення об'єктів. Щоб створити об'єкт класу “Car”, вкажіть ім'я класу, а потім ім'я об'єкта і використовуйте ключове слово **new**.

* **delegate** - це тип, який представляє собою посилання на методи з певним списком параметрів і типом повернення. Коли ви створюєте делегат, ви можете зв'язати його екземпляр з будь-яким методом із сумісною сигнатурою та типом повернення. Ви можете звернутися до методу (або викликати його) через екземпляр делегату. Делегати використовуються для передачі методів як аргументів іншим методам. Обробники подій - це ніщо інше, як методи, що викликаються через делегати. Ви створюєте користувацький метод, і клас, наприклад, елемент управління вікна, може викликати ваш метод, коли відбувається певна подія.

**Типи значень (Value Types)**

Типи-значення та типи-посилання - це дві основні категорії типів C#. Змінна типу-значення містить екземпляр типу. Цим вона відрізняється від змінної посилального типу, яка містить посилання на екземпляр типу. За замовчуванням, при присвоюванні, передачі аргументу в метод та поверненні результату методу значення змінних копіюються. У випадку зі змінними типу-значення копіюються екземпляри відповідних типів. Тип значення може бути одного з наступних двох типів:

* тип структури, який інкапсулює дані та пов'язану з ними функціональність.
* тип перерахування, який визначається набором іменованих констант і представляє вибір або комбінацію виборів.

**Ссилочні типи (Reference types)**

У мові C# є два типи: типи-посилання та типи-значення. Змінні посилальних типів зберігають посилання на свої дані (об'єкти), тоді як змінні типів-значень безпосередньо містять свої дані. У випадку посилальних типів дві змінні можуть посилатися на один і той самий об'єкт, тому операції над однією змінною можуть впливати на об'єкт, на який посилається інша змінна. У типах-значеннях кожна змінна має власну копію даних, і операції над однією змінною не можуть вплинути на іншу (за винятком змінних з параметрами *in, ref та out*).

Для оголошення типів-посилань використовуються наступні ключові слова:

* **клас (class)**
* **інтерфейс (interface)**
* **делегат (delegate)**
* **запис (record)**

У мові C# також передбачені наступні вбудовані типи посилань:

* **динамічний (dynamic)**
* **об'єкт (object)**
* **рядок (string)**

# **Оператори та вирази (C# operators and expressions)**

У мові C# передбачено ряд операторів. Багато з них підтримуються вбудованими типами і дозволяють виконувати базові операції зі значеннями цих типів. До таких операторів відносяться наступні групи:

* **Арифметичні оператори, які виконують арифметичні операції з числовими операндами**
* **Оператори порівняння, які порівнюють числові операнди**
* **Булеві логічні оператори, які виконують логічні операції з операндами типу bool**
* **Порозрядні оператори та оператори зсуву, які виконують порозрядні операції або операції зсуву з операндами цілих типів**
* **Оператори рівності, які перевіряють, чи рівні їхні операнди**

Як правило, ці оператори можна перевантажувати, тобто задавати поведінку оператора для операндів визначеного користувачем типу.

Найпростішими виразами C# є літерали (наприклад, цілі та дійсні числа) та імена змінних. Ви можете об'єднувати їх у складніші вирази за допомогою операторів. Операторний пріоритет і асоціативність визначають порядок виконання операцій у виразі. За допомогою круглих дужок ви можете змінити порядок обчислення, визначений пріоритетом оператора та асоціативністю. Як правило, вираз дає результат і може бути включений в інший вираз. Виклик методу **void** є прикладом виразу, який не дає результату.

**void** - Тип повернення методу (або локальної функції) **void** вказує на те, що метод не повертає значення. Також можна використовувати void як тип-посилання для оголошення вказівника на невідомий тип. Ви не можете використовувати void як тип змінної!

Ось деякі інші типи виразів, які надає мова C#:

* **Інтерпольовані рядкові вирази (Interpolated string expressions)**, які забезпечують зручний синтаксис для створення форматованих рядків:

var r = 2.3;

var message = $"The area of a circle with radius {r} is {Math.PI \* r \* r:F3}.";

Console.WriteLine(message);

// Output:

// The area of a circle with radius 2.3 is 16.619.

* **Лямбда-вирази (Lambda expressions)**, що дозволяють створювати анонімні функції:

int[] numbers = { 2, 3, 4, 5 };

var maximumSquare = numbers.Max(x => x \* x);

Console.WriteLine(maximumSquare);

// Output:

// 25

* **Вирази запитів(Query expressions)**, які дозволяють використовувати можливості запитів безпосередньо в C#:

var scores = new[] { 90, 97, 78, 68, 85 };

IEnumerable<int> highScoresQuery =

from score in scores

where score > 80

orderby score descending

select score;

Console.WriteLine(string.Join(" ", highScoresQuery));

// Output:

// 97 90 85

У наступній таблиці перераховано оператори C#, починаючи з найвищого пріоритету до найнижчого. Оператори в кожному рядку мають однаковий пріоритет.

* **Унарні (Unary)** - (+x; -x; ++x; –x; x++; !x; ~x ← логічна інверсія)
* **Мультиплікативні (Multiplicative)** - (x\*y; x/y; x%y)
* **Адитивні (Additive)** - (x+y; x-y)
* **Зміщення (Shift)** - (x<<y; x>>y, x>>>y)
* **Реляційні та рівності (Relational and type-testing)** - (x<y; x>y; x>=y; x<=y, is, as)
* **Рівностні (Equality)** - (x==y; x!=y)
* **Булеві (boolean) (логічні)** - (x&y ← множення; x|y ← додавання; x^y ← виключення or)
* **Умовні (Conditional)** - (x&&y ← and; x||y ← or)
* Присвоєння та лямбди (Assignment and lambda declaration) - (x = y; x += y; x -= y; x \*= y; x /= y; x %= y; x &= y; x |= y; x ^= y; x <<= y; x >>= y; x >>>= y; x ??= y; =>)