Тема : Основні типи даних мови C#

План

1. Елементи мови. Базова структура C#-програми.
2. Поняття типу даних, класифікація типів даних, їх застосування.
3. Опис змінних та констант.

Перша програма

Стандартний вигляд програми

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
class Program
{
static void Main(string[ ] args)
{



}
}
}
```

Простори імен

Простір імен (namespace) – це сукупність оголошень типів, співставлених та пов'язаних з певною назвою. За замовчуванням його назва співпадає з назвою проекту, яку задали при його створенні. У ньому оголошено власний тип – клас Program.

Простір імен оголошують так:

```
namespace TheNameOfNameSpace
```

Між фігурними дужками описують елементи простору імен – типи та класи. Простір імен є контейнером, у якому описують різні типи. Він забезпечує заданий програмістом вид групування коду, тому в одному просторі імен доцільно групувати логічно чи функціонально пов'язані типи.

Коментарі

Коментар – це довільний текстовий блок у програмі, який використовують для пояснень та приміток до відповідних ділянок коду. Коментарі дозволяють швидше зрозуміти роботу алгоритму та призначення різних його частин. Текст у коментарі **ігнорується компілятором** і не впливає на функціонування програми.

У C# використовують три види коментарів:

► Рядковий коментар - починається двосимвольним маркером "//" і продовжується до кінця рядка.

► Розмежований коментар - він починається двосимвольним маркером "/*" і закінчується двосимвольним маркером "*/". Розмежований коментар може також виділяти тільки частину одного рядка програми.

► Коментар для документації -починається трисимвольним маркером "///" і використовують для автоматизованого генерування програмної документації засобами Visual Studio.

Типи даних

Тип даних - це ***шаблон*** для створення структури даних. Тип даних визначає множину значень, до якої належать дані відповідного типу.

Тип визначають такі елементи:

► Назва типу. Вона використовується для створення об'єктів (екземплярів) типу.

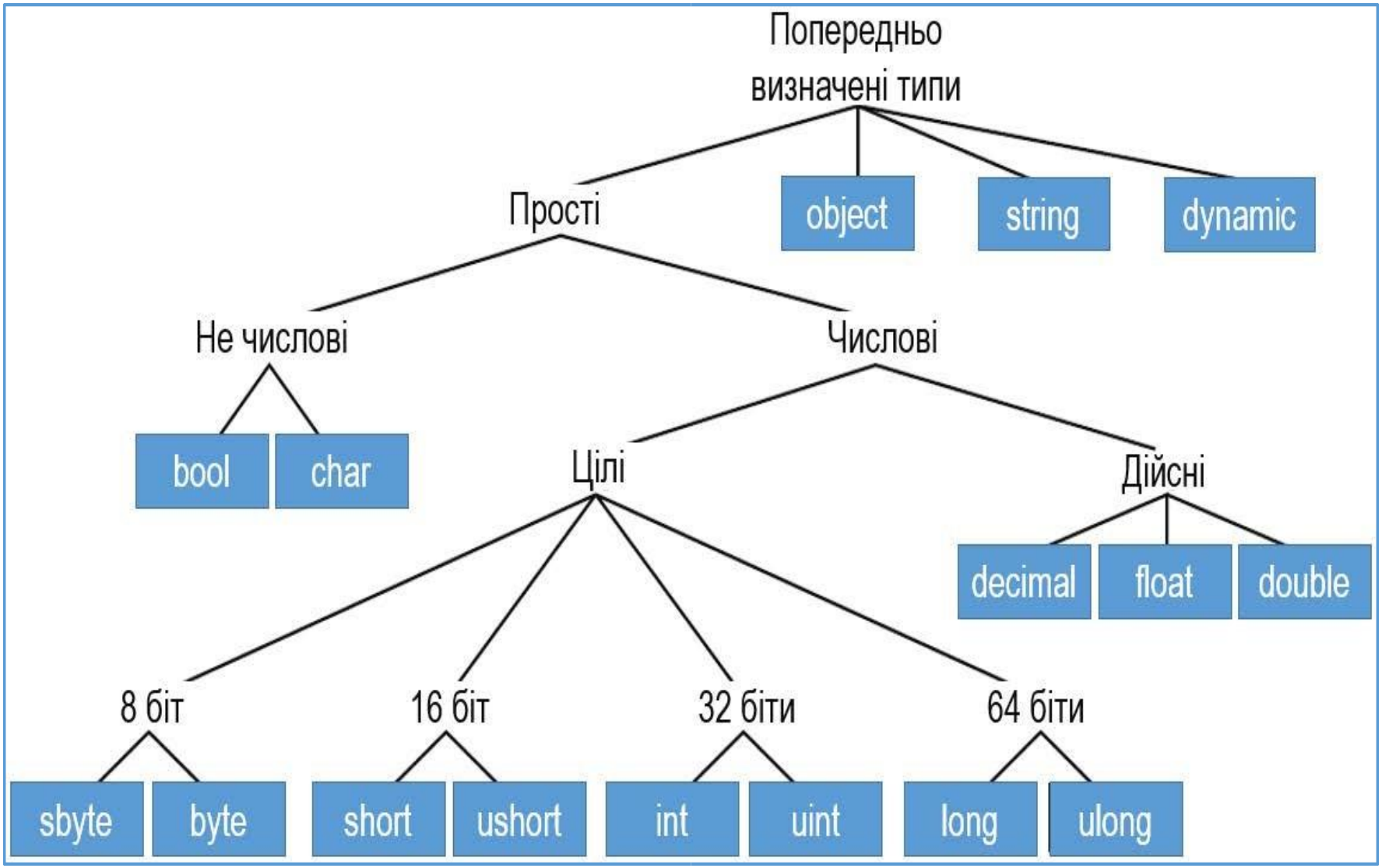
► Формат даних, які міститимуть інформацію про об'єкт.

► Призначення, поведінка та обмеження.

Наприклад, тип для позначення цілих чисел має назву int, у пам'яті займає 4 байти, і визначає поведінку цілого числа.

На основі певного типу можна створити багато об'єктів, чи екземплярів цього типу. Кожен елемент даних у програмі на C# є екземпляром типу, який визначений мовою програмування, BCL (чи іншою бібліотекою), або програмістом. При створенні екземпляра типу для нього виділяється потрібний обсяг пам'яті, структурований за шаблоном, що визначає тип.

Типи даних C#


Попередньо
визначені типи
Прості
object
string
dynamic
Не числові
Числові
bool
char
Цілі
Дійсні
decimal
float
double
8 біт
16 біт
32 біти
64 біти
sbyte
byte
short
ushort
int
uint
long
ulong

Типи даних

У C# визначено **16 типів:** 13 простих і 3 непростих. До простих типів відносять:

► **Числові типи:** різного розміру знакові та беззнакові цілі типи (byte, sbyte, short, ushort, int, uint, long, ulong); числа з плаваючою комою (float, double);

високоточний десятковий тип decimal, який, на відміну від типів з плаваючою комою, представляє десяткову частину числа точно.

► **Символьний тип char**, що представляє символ у форматі Unicode.

► **Логічний (булевий) тип bool**, який може приймати одне з двох значень: true (істина) чи false (хибність).

З кожним типом даних зв'язано своє унікальне *ім'я (ідентифікатор).*

Кожен із наведених типів є скороченням деякого системного типу. *Наприклад,* тип Int є альтернативною назвою структури System.Int32. В більшості випадків ці імена повністю рівноправні.

Типи даних

Тип C#	Опис	Діапазон	Тип .NET Framework
sbyte	8-бітове знакове ціле	-128…127	System.SByte
byte	8-бітове беззнакове ціле	0…255	System.Byte
short	16-бітове знакове ціле	-32768…32767	System.Int16
ushort	16-бітове беззнакове ціле	0…65535	System.UInt16
int	32-бітове знакове ціле	-2147483648… 2147483647	System.Int32
uint	32-бітове беззнакове ціле	0…4294967295	System.UInt32
long	64-бітове знакове ціле	-92233720368547758 08… 92233720368547758 07	System.Int64
ulong	64-бітове беззнакове ціле	0…18446744073709551615	System.UInt64
float	Число з плаваючою комою звичайної точності	-3.402×10^{-38}…3.402×10^{38}	System.Single
double	Число з плаваючою комою подвоєної точності	-1.797×10^{-308}…1.797×10^{308}	System.Double

Типи даних

До непростих типів входять:

► Тип object, який є базовим для всіх інших типів – як попередньо визначених, так і користувацьких.

► Рядковий тип string, що представляє рядок у форматі Unicode.

► Тип dynamic дозволяє пропускати перевірку типу під час компіляції. При цьому перевірка виконується під час виконання.

Тип C#	Опис	Тип .NET Framework
object	Базовий клас, від якого походять всі типи, включаючи прості типи	System.Object
string	Послідовність символів Unicode	System.String
dynamic	Використовується у збірках, написаних динамічними мовами	Немає відповідника

Типи даних

Всі попередньо визначені типи (крім типу dynamic) проектуються безпосередньо на типи .NET Framework. Насправді назви типів у C# є псевдонімами для типів .NET, які визначені у просторі імен System. Тому використовувати типи даних .NET у програмі також можна, але це без потреби заплутує код. В програмі на C# перевагу слід надавати "рідним" типам.
Але перевагу слід надавати першому способу, у якому використано тип даних C#.

3. Опис змінних та констант

Будь- яка програма має дані, з якими вона працює. Змінні призначені для зберігання даних в програмі. ***Змінна*** – це ім'я, яке під час виконання програми представляє конкретні дані у пам'яті. Кожна змінна має тип, який вказують при її оголошенні. При оголошенні змінної можна відразу присвоїти їй значення. Присвоєння змінній початкового значення називають її *ініціалізацією*. Для ініціалізації змінної під час оголошення дописують знак "=« та вказують потрібне значення змінної.

Загальне правило опису змінної:

- ***без початкової ініціініалізації***

<тип змінної> <ідентифікатор змінної>;

- ***з початковою ініціалізацією***

<тип змінної> <ідентифікатор змінної> = <значення>;

Наприклад:

int b=40; // Змінна b ініціалізовується значенням 40.

int a; // Без початкової ініціалізації

a=20;

3. Опис змінних та констант

Значення не може містити більше 255 символів та складатися з букв, цифр та знаку підкреслення, причому не може містити пропусків та знаків пунктуації.

Існують такі види змінних у C#:

► Локальні змінні. Вони містять тимчасові дані, доступні у певній ділянці програми. Такі змінні не є елементами типу.

► Поля. Вони містять дані, пов'язані з типом або з екземпляром типу, і є елементами типу

► Параметри. Це тимчасові змінні, які використовують для передачі даних від одного методу до іншого. Такі змінні не є елементами типу.

Оголошення змінної приводить до виділення для неї пам'яті.

3. Опис змінних та констант

Опис змінних з початковою ініціалізацією та без початкової ініціалізації називається явною типізацією змінних. В C# ми можемо використовувати також неявну типізацію, коли змінну можна оголосити з допомогою ключового слова var та вказати ідентифікатор змінної.
Наприклад,
var index=5; // int
var apple=“яблуко”; //змінна apple зберігає рядок, компілятор розглядає змінну як тип string
При компіляції компілятор автоматично на основі значення змінної виведе її тип.

Зауваження!Тип змінної не можна міняти під час виконання програми. Такий підхід називають *статичною типизацією*.

Опис констант

Константи - незмінні значення (іноді їх ще називають літералами). Літерали можна передавати змінним в якості значення. Константи бувають логічними, цілочисельними, речовими, символьними і малими. Ще один окремий літерал являє ключове слово null.

Загальне правило опису:

const <тип константи> <ідентифікатор констати> = <значення>;

Приклади:

const int a = 10; // константа цілого типу

const float b = 2.5; // константа дійсного типу

Опис констант

При роботі з інформацією, яка представляє реальні дані, часто виникає необхідність показати, що змінна не містить коректних даних. З цією метою використовують спеціальне значення null. Це значення не тотожне значенню 0 (нуль). Воно означає відсутність даних.

Типам значень не можна присвоювати значення null. Але C# підтримує модифікатор "?", який використовують для оголошення змінної, що може приймати значення null.

```
int? i = null; i = 10;
Console.WriteLine("i = {0}", i);
```

При оголошенні змінної з модифікатором типу "?" насправді створюється змінна зовсім іншого типу – екземпляр структури System.Nullable<>, яка має свої властивості та методи.

Тема : Вирази та операції.

План

1. Арифметичні оператори та вирази. Бінарні та унарні операції.
2. Логічні операції. Операції побітового зсуву.
3. Операції присвоєння.
4. Математичні функції.

ВИРАЗИ ТА ОПЕРАЦІЇ

Оператор – це символ (або група символів), які представляють операцію, що повертає єдиний результат. *Операнд* – це елемент вхідних даних для операції.

Оператор:

- ► Отримує операнди на вході.
- ► Виконує дію.
- ► Повертає значення результату.

Виразом називають композицію операндів та операцій над ними. Кожен вираз визначає результат певного типу. В залежності від типу результату вирази, найчастіше, поділяють на ***арифметичні, логічні та символьні.***

Арифметичним виразом називають вираз, в результаті обчислення якого одержуємо числове значення.

Бінарні операції

Арифметичні оператори

Оператор	Операція	Приклад
+	додавання	z=x+y
–	віднімання	z=x-y
*	множення	z=x*y
/	ділення	z=x/y
%	остача від ділення	z=x%y

Унарні операції

Окрім бінарних операцій в арифметичному виразі можуть бути присутні також ***унарні операції*** «+», «–» та ***операції інкременту*** «++» і декременту «– –». Унарний мінус використовується для зміни знаку. Операції інкременту і декременту використовують для збільшення та зменшення значення змінної на одиницю. Ці операції можуть вживатися у різних формі.

Оператор	Назва	Опис
++	Префіксний інкремент: ++i	Збільшує значення змінної на 1 і записує його у цю змінну. Повертає нове значення змінної.
	Постфіксний інкремент: i++	Збільшує значення змінної на 1 і записує його у цю змінну. Повертає попереднє значення змінної, до інкременту.
--	Префіксний декремент: --i	Зменшує значення змінної на 1 і записує його у цю змінну. Повертає нове значення змінної.
	Постфіксний декремент: i--	Зменшує значення змінної на 1 і записує його у цю змінну. Повертає попереднє значення змінної, до

Операції побітового зсуву

Операції побітового зсуву застосовуються до цілочислових операндів. Вони зсувають двійкове представлення першого операнда вліво або вправо на кількість двійкових розрядів, яка задається другим операндом.

Правило запису:

- При зсуві вліво (<<) вільні біти обнуляються.
- При зсуві вправо (>>) вільні біти заповнюються нулями, якщо перший операнд беззнакового типу

(логічний зсув), і знаковим розрядом − в протилежному випадку (арифметичний зсув).

Побітові та логічні оператори

Оператор	Назва
&	Побітове "І"
|	Побітове "АБО"
^	Побітове "Виключаюче АБО"
~	Побітове "НЕ"
<<	Зсув бітів вліво
>>	Зсув бітів вправо

Оператори присвоєння

Оператори присвоєння обчислюють значення виразу справа від оператора і присвоюють це значення змінній зліва від оператора.

Оператор	Опис
=	Базовий простий оператор присвоєння. Результат виразу справа від оператора присвоюється змінній зліва
	Складні оператори присвоєння
*=	y *= x еквівалентно до y = y * x
/=	y /= x еквівалентно до y = y / x
%=	y %= x еквівалентно до y = y % x
+=	y += x еквівалентно до y = y + x
-=	y -= x еквівалентно до y = y - x
<<=	y <<= x еквівалентно до y = y << x
>>=	y >>= x еквівалентно до y = y >> x