### Коментарі

*Коментар* – це довільний текстовий блок у програмі, який використовують для пояснень та приміток до відповідних ділянок коду. Коментарі, особливо у великих за обсягом та складних програмах, дозволяють швидше зрозуміти роботу реалізованого 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. Він починається двосимвольним маркером "//" і продовжується до кінця рядка. Приклад використання такого коментаря приведено у лістингу 2.1 (рядок 9).

► Розмежований коментар. Він починається двосимвольним маркером "/\*" і закінчується двосимвольним маркером "\*/". Весь текст між цими маркерами (він може містити багато рядків) вважається коментарем. Розмежований коментар може також виділяти тільки частину одного рядка програми.

► Коментар для документації. Такий коментар починається трисимвольним маркером "///" і містить XML-текст, який використовують для автоматизованого генерування програмної документації засобами

Visual Studio.

Приклади всіх згаданих коментарів приведено у лістингу 2.2. Для покращеного сприйняття тексту програми редактор коду Microsoft Visual Studio виділяє тексти коментарів іншим кольором (найчастіше – зеленим).

|  |  |
| --- | --- |
| Лістинг 2.2. Різні види коментарів | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | // У рядку 1 приведено зразок однорядкового коментаря int i ; // Так можна закінчувати рядок програми, // пояснюючи призначення коду в ньому  /\* Це приклад розмежованого коментаря, який може займати кілька рядків коду. Коментар будь-якого виду компілятор ігнорує \*/    // Далі приклад розмежованого коментаря в частині рядка // Наступний рядок для компілятора буде таким: int j; int /\*i,\*/ j; // Нижче приклад коментаря для документації:  /// <summary> /// Цей клас призначений для ...  /// <\summary> class Department { ... |

### Типи і змінні

Із загального погляду програма на C є сукупністю функцій та типів даних, а програма на C++ – сукупністю функцій та класів. У той же час програму на C# можна розглядати як сукупність оголошень типів. Зокрема, програма з лістингу 2.1 містить оголошення типу (класу) Program з методом Main. Таким чином, еволюція мов від C до C# ілюструє все більшу вагу об'єктноорієнтованих принципів у програмуванні. Для розуміння програми на C# важливо розуміти, чим є типи і як з ними працювати.

#### Що таке тип

В загальному тип можна розглядати як *шаблон* для створення структури даних. Він сам по собі не є структурою даних, але задає характеристики об'єктів, створених за цим шаблоном. *Об'єктом*, або *екземпляром*, називають структуру даних, створену за шаблоном типу.

Тип визначають такі елементи:

► Назва типу. Вона використовується для створення об'єктів

(екземплярів) типу.

► Формат (устрій) 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.

#### Елементи даних та функціональні елементи типу

Деякі типи, які називають *простими типами* (int, short, long), зберігають єдину складову даних. Інші типи можуть зберігати багато складників даних. Наприклад, *масив* – це тип, який містить багато складників одного і того ж типу. До окремих його елементів можна отримати доступ через спеціальне число, яке називають індексом.

Але можуть бути і типи, які містять у складі дані різних типів. Такі складники типу називають *членами* типу, або *елементами* типу. На відміну від масиву, кожен елемент типу має свою назву, за якою до нього і звертаються.

Є два види елементів типу: елементи даних та функціональні елементи. Елементи даних зберігають дані, які стосуються екземпляра типу або типу в цілому. Функціональні елементи містять виконавчий код і визначають поведінку об'єктів цього типу.

Тип даних, який може містити іменовані елементи даних та функціональні елементи, називають *класом*.

#### Попередньо визначені типи

У C# визначено 16 типів, серед них 13 простих і 3 непростих (рис. 2.5). Назви всіх попередньо визначених типів містять тільки малі символи. Прості типи (див. таблицю 2.5) формують такі групи:

► 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.

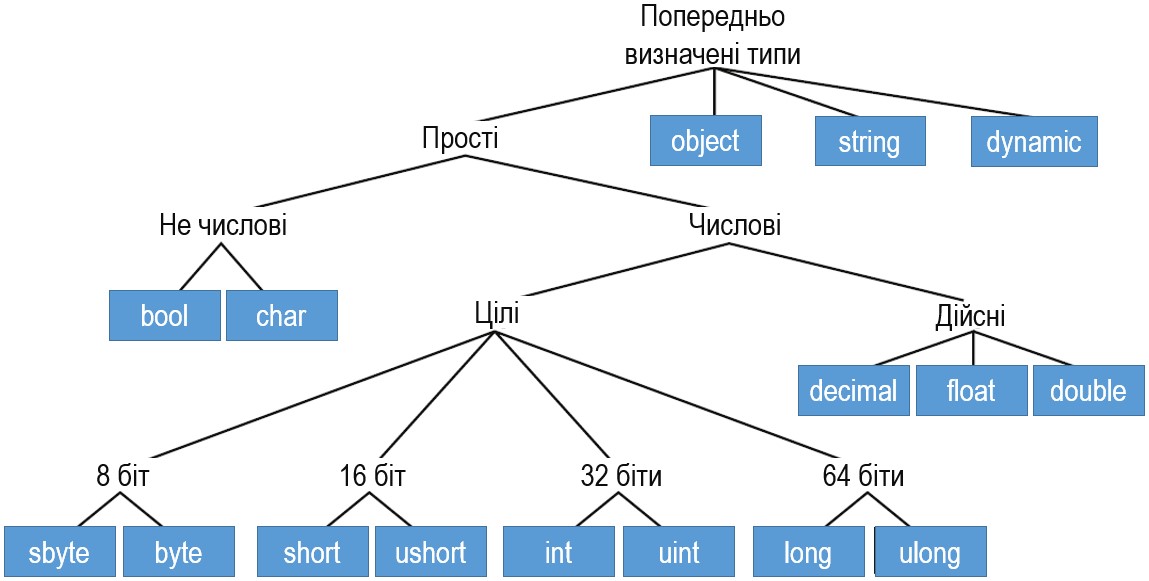
► Символьний тип char, що представляє символ у форматі Unicode.

► Логічний (булевий) тип bool, який може приймати одне з двох значень: true (істина) чи false (хибність)[[1]](#footnote-1).

Рис.

2.5

. Попередньо визначені типи C#



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 інструкції, в якій він застосовується. Замість цього перевірка виконується під час виконання.

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.

Таким чином, оголошення змінної Count у приведеному коді рівнозначні:

using System; ... int Count; // Тип даних C# Int32 Count; // Той же тип даних, але .NET Framework

Але перевагу слід надавати першому способу, у якому використано тип даних C#.

У таблиці 2.5 описано прості попередньо визначені типи та приведено їх відповідники з .NET Framework. Мінімальне та максимальні значення для типів, які представляють дійсні числа, подано із заокругленням. Таблиця 2.6 містить опис інших попередньо визначених типів C#.

Таблиця №2.5. Попередньо визначені прості типи C#

|  |  |  |  |
| --- | --- | --- | --- |
| **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 | -9223372036854775808…  9223372036854775807 | System.Int64 |
| **ulong** | 64-бітове беззнакове ціле | 0…18446744073709551615 | System.UInt64 |
| **float** | Число з плаваючою комою звичайної точності | -3.402×10-38…3.402×1038 | System.Single |
| **double** | Число з плаваючою комою подвоєної точності | -1.797×10-308…1.797×10308 | System.Double |
| **decimal** | Десятковий тип | ±7.923×1028 | System.Decimal |
| **bool** | Логічний тип | false, true | System.Boolean |
| **char** | Символ Unicode |  | System.Char |

Таблиця №2.6. Попередньо визначені непрості типи C#

|  |  |  |
| --- | --- | --- |
| **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 програмування | Немає відповідника |

#### Типи, які визначає користувач

На основі попередньо визначених типів можна створювати власні типи, призначені для опису структур даних, потрібних за алгоритмом програми. Є такі види типів, які може визначати користувач:

► Класи (class).

► Структури (struct).

► Масиви (array).

► Переліки (enum).

► Делегати (delegate).

► Інтерфейси (interface).

Тип створюють, використовуючи синтаксис оголошення типу. Він містить інформацію про вид типу даних (клас, масив тощо), його назву та оголошення всіх елементів типу (за винятком масивів та делегатів, які не мають іменованих елементів).

Після того, як тип оголошено, можна створювати та оголошувати об'єкти цього типу. Таким чином, користувацькі типи застосовують у програмі так же, як і попередньо визначені, за винятком того, що їх спочатку слід оголосити.

Перелічені користувацькі типи детально розглянемо у наступних розділах.

1. На відміну від C та C++, у C# числове значення не має логічної інтерпретації. [↑](#footnote-ref-1)