

3.3. Константи

Константи являють собою фіксовані значення, що не можуть змінюватися впродовж виконання всієї програми.

Спосіб визначення кожної константи залежить від її типу. Константи мови C++ слід поділяти на літеральні та типізовані.

Літеральна константа — це лексема, що являє собою зображення фіксованого числового, рядкового або символного значення. Такі константи бувають **цілі, дійсні, символні та рядкові** (табл. 3.2).

Таблиця 3.2

Літеральні константи мови C++

Константа	Формат	Приклади
Ціла	<i>Десятковий:</i> послідовність десяткових цифр (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) , що починається не з нуля, якщо це не число нуль <i>Вісімковий:</i> нуль, за яким розташовані вісімкові цифри (0, 1, 2, 3, 4, 5, 6, 7) <i>Шістнадцятковий:</i> 0x чи 0X, за яким розташовані шістнадцяткові цифри (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)	9, 0, 217925 02, 050, 07245 0x1B9, 0X00FF
Дійсна	<i>Десятковий:</i> [цифри].[цифри] <i>Експоненціальний:</i> [цифри][.][цифри]{E e}{+ -}[цифри]	9.7, .001, 87. 0.7E6, .15e-3, 9.2, 920 e-2, 92.E-1, .92E1
Символьна	Один чи два символи, що подаються в апострофах	'A', 'ю', '* ', 'db', '\0', '\n', '\012', '\x07\x07'
Рядкова	Послідовність символів, що подаються в лапках	"RESULT", "\t sum_s=\0x5\n"

Цілі константи можуть бути *десятковими, вісімковими та шістнадцятковими*.

Довгі цілі константи (**long**) мають літеру **l** або **L** в кінці, наприклад: 32768L; 0777777l; 0XFL. Для завдання константи, без знака (**unsigned**) застосовується літера **u** (**U**), наприклад 65535u. Довгі константи без знака записуються з використанням двох літер відразу: (**ul, UL**) або (**lu, LU**).

Дійсні числа у мовах програмування мають дві форми подання: десяткову (природну) та експоненціальну (показникову).

Десяткова форма дійсного числа — це звичайний десятковий формат запису дійсного числа, тільки частина дійсного числа відділяється від дробової крапкою, а не комою, наприклад 10.123, 1.0123, 1012.3, 0.0010123.

Експоненціальна форма дійсного числа використовується для запису дуже великих або дуже малих чисел, для яких задавати зайві нулі не зовсім зручно, наприклад: 1 .0123*10²⁰, 1.0123*10⁻¹⁰. У цій формі запису числа можна виділити такі основні характеристики: знак числа, мантису числа, знак порядку та порядок числа. Зазначені характеристики дійсного числа зберігаються у пам'яті комп'ютера. Число у показниковій формі може бути представлено, наприклад, так: 1.0123E-10.

Мантиса записується ліворуч від знака експоненти (*E* чи *e*), порядок — праворуч. Символ **E** (**e**) означає основу степеня 10, і компілятор розпізнає цей запис як форму представлення дійсного числа. Символ пропуску всередині числа не допускається, а для відділення цілої частини від дробової використовується не кома, а крапка. При додатних значеннях числа і мантиси знак «+» можна не вказувати.

Як десяткова, так і експоненціальна форми запису допускають відсутність або цілої частини, або дробової, але не двох одразу.

За замовчуванням всі дійсні константи мають тип **double** — подвійну точність, що найчастіше займає в пам'яті 64 біти, тобто 8 байтів. Але у випадку, якщо програміста не влаштовує тип за замовчуванням, його можна вказати явно за допомогою спеціальних літер. Так, додавши літеру **f** чи **F**, константі надають дійсний тип **float** зі звичайною точністю, наприклад, 8.5f. Якщо в представленні константи використовується літера **L** чи **l**, то вона має тип **long double**.

Зображення від'ємної цілої чи дійсної константи вважається константним виразом, що складається зі знака унарної операції зміни знака (-) та константи, наприклад: -273, -2730.e-1, -273L.

Символьні константи мають один або два символи, що подаються в апострофах. Односимвольні константи займають у пам'яті один байт і мають стандартний тип **char** (*character-символ*). Двосимвольні константи займають два байти і мають тип **int**. Символьні константи мають цілий тип і їх можна використовувати як цілочислові операнди у виразах.

Заслуговують уваги послідовності, що починаються зі знака «\», їх називають **керуючими** або **escape-послідовностями**. Символ зворотної косої риски «\» використовується для запису кодів, що не мають графічного зображення, для запису символів, а також для виведення символьних констант, якщо їх коди задані у вісімковому та шістнадцятковому вигляді (*табл. 3.3*).

Таблиця 3.3

Керуючі послідовності мови C++

\a	звуковий сигнал
\b	повернення на крок
\f	переведення сторінки (формату)
\n	новий рядок
\r	повернення каретки
\t	горизонтальна табуляція
\v	вертикальна табуляція
\\	символ «\» — зворотна коса риска
\'	символ «'» — апостроф
\"	символ «"» — лапки
\0	нуль-символ
\?	знак питання
\0ddd	вісімковий код символу
\xddd	шістнадцятковий код символу

Рядкова константа (рядковий літерал) — це послідовність символів, що подається в лапках (тобто в символах «"») і зберігається у неперервній ділянці пам'яті, наприклад: **“Це рядковий літерал”**. У кінець кожного рядкового літералу компілятором додається нуль-символ, що

представляється керуючою послідовністю «\0». Тому довжина рядка завжди на одиницю більше кількості символів у його записі. Таким чином, порожній рядок (” “) має довжину 1 байт. Слід звернути увагу на різницю між рядком з одного символу, наприклад, “С” і символічною константою ‘С’. Порожня символічна константа неприпустима.

Керуючі послідовності можуть також застосовуватись у рядкових константах. Так, якщо всередині рядка потрібно записати лапки, то перед ними слід розташувати зворотну косу риску («\»), за якою компілятор відрізняє їх від лапок, що обмежують рядок:

“Книга має назву \”Мова програмування С++\” “.

Рядки, що записані у програмі підряд або через символи пропуску, при компіляції конкатенуються («склеюються»). Тобто послідовність двох рядків

“Навчаючи інших, ми вчимося самі.”

“Успіх — це встигнути.”

цілком еквівалентна рядку:

“Навчаючи інших, ми вчимося самі. Успіх — це встигнути.”

Довгу рядкову константу можна розмістити також на декількох рядках. У цьому випадку ставиться зворотна коса риска і натискається клавіша **Enter**. Наприклад:

**“Комп’ютерна програма виконує те, **
**що ви їй наказали виконати, а не те, **
що ви хотіли, щоб вона виконувала.”

Поняття та приклади оголошення **типізованої константи**, тобто константи, що використовується як змінна, значення якої не може бути змінене після ініціювання, розглянуті вище.

Існує інша можливість задання констант — з використанням директиви препроцесора **#define**, при цьому оголошення має вигляд:

#define ім’я константи значення константи

і наприкінці такого запису символ «;» не ставиться, тобто:

#define max 65532
#define km 1000.

Директива **#define** визначає ідентифікатор (**ім’я константи**) і послідовність символів (**значення константи**), яка замінює ідентифікатор у тексті програми.

Нульовий покажчик (NULL-покажчик) — єдина неарифметична константа мови С++.

При застосуванні великої кількості логічно взаємозалежних констант С++ доцільно користуватися **константами перелічення**. Тип перелічення має вигляди:

enum {список іменованих констант};

— неіменоване перелічення,

enum [ім’я] {список іменованих констант};

— іменоване перелічення.

де **enum** — службове слово (**enumerate**—перелічувати);

ім'я — ім'я списку констант;

список іменованих констант — розділена комами послідовність ідентифікаторів або іменованих констант вигляду:

ім'я константи = значення константи.

Наприклад:

```
enum {Anton, Ivan, Piter};
```

```
enum Months {January = 1, February, Marth, April, May, June, July, August, September, October,  
November, December};
```

Якщо значення константи перелічення не визначено, то воно на одиницю більше значення попередньої константи. За замовчуванням перша константа має значення **0**. Тоді у першому прикладі константи одержать значення: **Anton = 0, Ivan = 1, Piter = 2**, а у другому — значення: **January = 1, February = 2, Marth = 3** тощо. Іменовані перелічення задають унікальний цілочисловий тип і можуть використовуватися як специфікації типу для визначення змінних.