

## 3.2. Змінні

Кожна програма потребує виконання різноманітних обчислень, для здійснення яких використовуються вирази, що складаються з операндів, знаків операцій і дужок. Операнди задають дані для обчислень, а операції задають дії, які необхідно виконати над цими даними. Операнд є, у свою чергу, виразом, що в окремому випадку може бути константою або змінною.

**Змінна** — це іменована область пам'яті, у якій зберігаються дані визначеного типу. Змінна має ім'я, розмір та інші атрибути, такі як видимість, час існування тощо. Ім'я змінної служить для звертання до області пам'яті, у якій зберігається її значення. *Перед використанням будь-яка змінна повинна бути описана*, при цьому для неї резервується деяка область пам'яті, розмір якої залежить від конкретного типу змінної. Під час виконання програми змінна може приймати різні значення.

Наведемо загальний вигляд опису змінних:

```
[клас пам'яті] [const] тип ім'я [ініціювання];
```

де необов'язковий клас пам'яті може приймати одне зі значень — **auto**, **extern**, **static** чи **register** (у посібнику при описі синтаксису об'єктів програмування необов'язкові частини синтаксичних конструкцій мови подано у квадратних дужках «[ ]»);

модифікатор **const** вказує, що змінна не може змінювати своє значення, у цьому випадку її називають **типизованою (іменованою) константою** або просто **константою**;

**ініціювання** — це присвоювання змінній при описі початкового значення, яке записується зі знаком рівності — = **значення** або в круглих дужках — (**значення**). Зазначимо, що константа *повинна бути ініційована при описі*. Один оператор може містити опис декількох змінних одного типу, розділяючи їх комами, наприклад:

```
const int n = 20, m = 5, k = 4; — ініціювання констант n, m, k цілого типу;  
float h = 17.5, d(5.5), sum; — опис дійсних змінних h, d, sum, ініціювання h і d;  
char sf = 'f', st[ ] = "Мудрість лише в істині."; — ініціювання символічних змінних.
```

Якщо **тип** значення, що ініціюється, не збігається з типом змінної, то виконуються перетворення типу. Кожна змінна повинна мати своє ім'я, причому в одному блоці не може бути двох змінних з однаковим ім'ям.

**Областю дії ідентифікатора змінної** є частина програми, в якій його можна використовувати для доступу до зв'язаної з ним області пам'яті. Залежно від області дії змінна може бути локальною або глобальною.

**Локальна змінна** визначена всередині блока (нагадаємо, що блок розташований між фігурними дужками). Область її дії обмежена початком опису змінної та кінцем блока, включаючи усі вкладені блоки. Змінна, визначена поза будь-яким блоком, називається глобальною, і областю її дії вважається файл, у якому вона визначена від початку опису до його кінця.

**Клас пам'яті** визначає час існування та область видимості програмного об'єкта, тобто змінної. Якщо клас пам'яті не зазначений явно, то він визначається компілятором, виходячи, з контексту оголошення.

**Час існування** змінної може бути постійним (протягом виконання програми) і тимчасовим (протягом виконання блока).

**Область видимості** ідентифікатора називається частина тексту програми, з якої можна здійснити звичайний доступ до зв'язаної з ідентифікатором області пам'яті. Найчастіше область видимості збігається з областю дії. Винятком є ситуація, коли у вкладеному блоці описана змінна з таким же ім'ям. У цьому випадку зовнішня змінна у вкладеному блоці невидима, хоча він і входить до її області дії. Проте до цієї змінної, якщо вона глобальна, можна звернутися, застосовуючи операцію доступу до області видимості — "::".

Клас пам'яті задають такі специфікатори:

- **auto** — автоматична змінна, для якої пам'ять виділяється у стеку і за необхідності ініціюється кожного разу при виконанні оператора, що містить її визначення. Звільнення пам'яті відбувається при виході з блока, де описана змінна. Час її існування — з моменту опису до кінця виконання блока. Для глобальних змінних цей специфікатор не використовується, а для локальних він приймається за замовчуванням, тому задавати його явно великого сенсу немає;
- **extern** означає, що змінна визначена в іншому місці програми (в іншому файлі або далі по тексті) і використовується для створення змінних, доступних в усіх модулях програми, де вони оголошені. При ініціюванні змінної у тому ж операторі, специфікатор **extern** ігнорується;
- **static** — статична змінна, що має постійний час існування. Вона ініціюється один раз при першому виконанні оператора, що містить визначення змінної. Залежно від розташування оператора, описані статичні змінні можуть бути глобальними і локальними. Глобальні статичні змінні видимі тільки у тому модулі, в якому вони описані;
- **register** — аналогічний до специфікатора **auto**, але пам'ять виділяється по можливості в регістрах процесора і за відсутності такої можливості у компілятора змінні обробляються як **auto**.

Наведемо фрагмент програми з використанням розглянутих вище понять:

```
int d;           //1 - глобальна змінна d
int main()
{
  int b;        //2 - локальна змінна b
  extern int y; //3 - змінна у визначена в іншому місці програми
  static int s; //4 - локальна статична змінна s
  d = 1;        //5 - присвоєння значення глобальній змінній
  int d;        //6 - локальна змінна d
  d = 10;       //7 - присвоєння значення локальній змінній
  ::d = 3;      //8 - присвоєння значення глобальній змінній
  return 0;
}
int y = 4;      // 9 - визначення і ініціалізація змінної у
```

У цьому прикладі глобальна змінна **d** визначена поза всіма блоками. Пам'ять для неї виділяється в сегменті даних на початку роботи програми, областю дії є вся програма. Область видимості — вся програма, крім рядків 6-8, тому що в першому з них визначається локальна змінна з тим же ім'ям, область дії якої починається з початку її опису і закінчується при виході з блока. Змінні **b** і **s** — локальні, область їх видимості — блок, але час існування різний: пам'ять під **b** виділяється в стеку при вході у блок і звільняється при виході з нього, а змінна **s** розташована у сегменті даних та існує

увесь час роботи програми. Якщо початкове значення змінних явно не задається, компілятор присвоює глобальним і статичним змінним нульове значення відповідного типу. Автоматичні змінні не ініціюються. *Початкове ініціювання змінних не є обов'язковим, проте все ж його бажано здійснювати.*

Опис змінної може виконуватися у формі оголошення або визначення. Оголошення інформує компілятор про тип змінної і класи пам'яті, а **визначення** містить, крім цього, вказівку компілятору про виділення пам'яті відповідно до типу змінної. У C++ більшість оголошень є одночасно і визначеннями (у наведеному вище програмному фрагменті тільки опис **extern int y**; є оголошенням, але не визначенням). *Змінна може бути оголошена багаторазово, а визначена тільки в одному місці програми*, оскільки оголошення тільки описує властивості змінної, а визначення зв'язує її з конкретною областю пам'яті.

Розглянемо далі основні типи змінних.

**Цілі змінні** (типу **int, long, short**) необхідні для збереження цілих значень і можуть бути знаковими і беззнаковими. Знакові змінні застосовують для подання як додатних, так і від'ємних чисел, при цьому один біт (найстарший) виділяється під знак. Для оголошення беззнакової змінної, тобто змінної, що приймає тільки додатні значення, необхідно використовувати ключове слово **unsigned**. За замовчуванням будь-який цілий тип вважається знаковим, і тому немає потреби у використанні ключового слова **signed**.

**Символьний тип** даних **char** застосовується у випадку, коли змінна містить інформацію про код ASCII або для побудови таких більш складних конструкцій, як рядки, символьні масиви тощо. Дані типу **char** також можуть бути знаковими і беззнаковими.

**Змінна типу bool** займає 1 байт і використовується, насамперед, у логічних операціях, тому що може приймати значення **0 (false** — «неправда») або відмінне від нуля (**true** — «істина»). У випадку перетворення до цілого типу **true** має значення 1.

Стандарт C++ визначає три типи даних для збереження *дійсних значень змінних*: **float, double** та **long double** (типи з *плаваючою крапкою*). Тип **float**, як правило, використовують для збереження не дуже великих дробових чисел.

**Змінна типу void** не має значення, оскільки множина значень цього типу порожня. Такі змінні необхідні для узгодження синтаксису. Тип **void** використовується для визначення функцій, що не повертають значення, для вказівки порожнього списку аргументів функції, а також як базовий тип для покажчиків і в операції приведення типів. Наприклад, якщо немає потреби у використанні поверненого значення функції, перед ім'ям функції ставиться тип **void**:

```
void minmax(int*x, int k, int*min, int&max);
```