

ЛЕКЦІЯ 4

1. ВИРАЗИ
2. ОПЕРАЦІЇ
3. ДВОВИМІРНІ МАСИВИ. АЛГОРИТМ ОБРОБКИ.

ВИРАЗИ

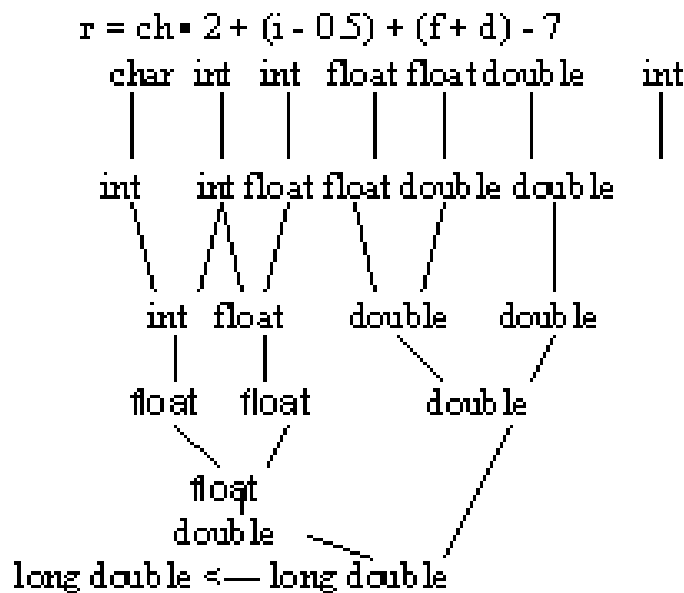
Вираз в мові C - це деяка допустима комбінація змінних, констант і операцій. Кожен вираз в мові C приймає будь-яке значення. У виразах допустимо використання змінних різного типу. Правила мови C, які використовуються для автоматичного приведення типів при обчисленні арифметичного виразу, такі:

1. Всі змінні типу `char` і `short int` перетворюються в `int`, всі змінні типу `float` перетворюються в `double`.
2. Для будь-якої пари операндів: якщо один з операндів `long double`, то й інший перетвориться в `long double`; якщо один з операндів `double`, то й інший перетвориться в `double`; якщо один з операндів `long`, то й інший перетвориться в `long`; якщо один з операндів `unsigned`, то й інший перетвориться в `unsigned`.
3. В операторі присвоювання кінцевий результат приводиться до типу змінної в лівій частині оператора присвоювання, при цьому тип може як підвищуватися, так і знижуватися.

Приклад перетворень типів змінних при обчисленні виразів. Нехай визначені змінні таких типів:

char ch; int i; float f; double d; long double r;

При обчисленні виразу відбудуться наступні автоматичні перетворення типів:



Тип результату обчислення виразу можливо змінити, використовуючи конструкцію приведення (casts), що має такий вигляд:

(Тип) вираз

Тут "тип" - один з стандартних типів даних мови C.

Наприклад, якщо необхідно, щоб результат ділення змінної `x` типу `int` на 2 був типу `float`, потрібно написати

`(Float) x / 2`

Пробіли та дужки у виразах можна розставляти так, як необхідно за логікою. Це робиться для зручності читання програми. При компіляції зайві пробіли просто ігноруються.

ОПЕРАЦІЇ

Мова С включає багато операцій. Знак операції - це символ або комбінація символів, які повідомляють компілятору про необхідність провести певні арифметичні, логічні або інші дії. Повний список знаків операцій мови С наведено нижче.

[]	()	.	->	++	--
&	*	+	-	~	!
sizeof	/	%	<<	>>	
<	>	<=	>=	==	!=
^		&&		?:	=
*=	/=	%=	+=	-=	<<=
>>	&=	^=	=	,	

Для кожної операції мови C визначено кількість операндів:

- один операнд - унарна операція, наприклад, унарний мінус $-x$, що змінює знак;
- два операнда - бінарна операція, наприклад, операція додавання $x + y$ або віднімання $x - y$;
- три операнда - операція умова $?:$, Така операція тільки одна.

Кожна операція може мати тільки певні типи операндів. Наприклад, кожна бінарна операція має певний порядок виконання: зліва направо або справа наліво. Наприклад, операція додавання виконується зліва направо, а операція присвоювання виконується справа наліво.

Нарешті, кожна операція має свій пріоритет. Так, операція додавання має більш низький пріоритет перед операцією множення. У той же час операція додавання має більш високий пріоритет перед операцією присвоювання. Як правило, унарні операції мають більш високий пріоритет, ніж бінарні.

АРИФМЕТИЧНІ ОПЕРАЦІЇ

До арифметичних операцій мови C відносяться:

- віднімання й унарний мінус;
- + додавання;
- * множення;
- / ділення;
- % ділення по модулю;
- ++ збільшення на одиницю (increment);
- зменшення на одиницю (decrement).

Операції додавання, віднімання, множення і ділення діють так само, як і в більшості інших алгоритмічних мов. Вони можуть застосовуватися до всіх скалярних типів даних. Операції виконуються зліва направо, тобто спочатку обчислюється вираз лівого операнда, потім вираз, що стоїть праворуч від знака операції. Якщо операнди мають один тип, то результат арифметичної операції має той же тип.

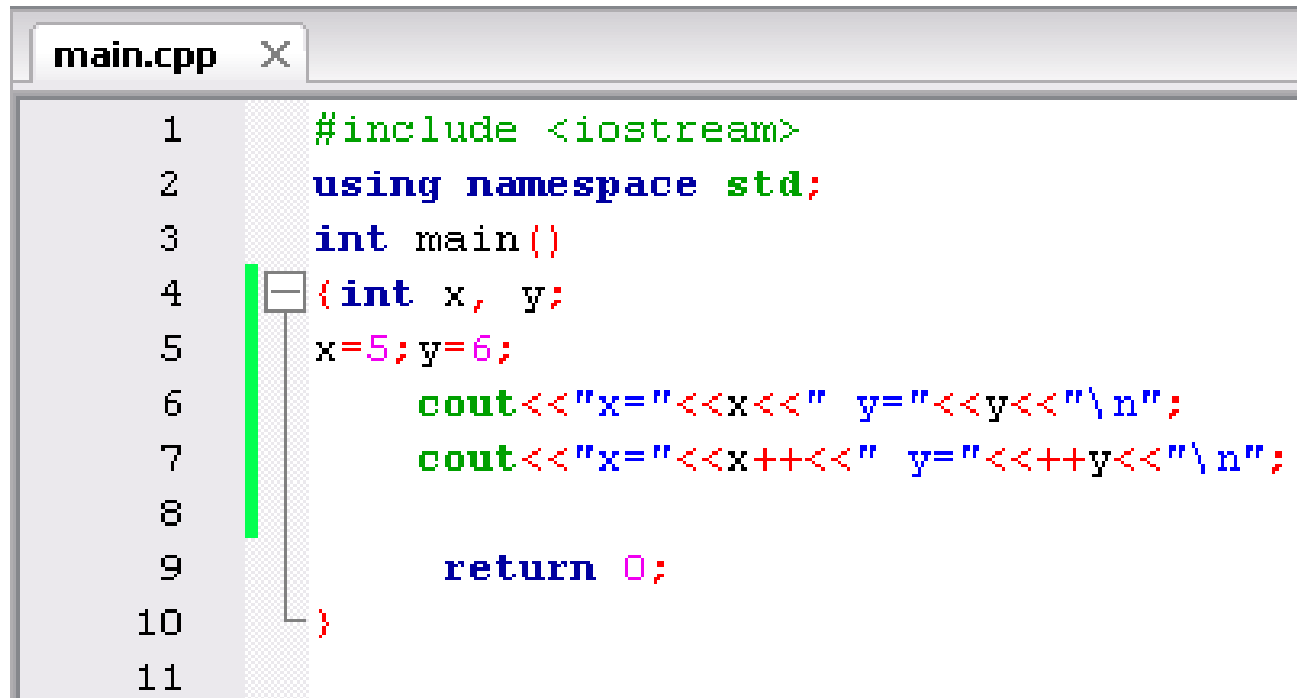
Тому, коли операція ділення / застосовується до цілих змінних або символьних змінних, остача відкидається. Так, $11/3$ дорівнюватиме 3, а вираз $1/2$ дорівнюватиме нулю.

Операція ділення по модулю % дає остачу від цілочисельного ділення. Операція % може застосовуватися тільки до цілочисельних змінних.

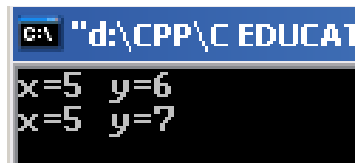
Мова C надає користувачеві ще дві дуже корисні операції, специфічні саме для мови C. Це унарні операції ++ і --. Операція ++ додає одиницю до операнду, операція -- віднімає одиницю з операнда. Обидві операції можуть слідувати перед операндом або після операнда (префіксная і постфіксная форми). Три написані нижче оператора дають один і той же результат, але мають відмінність при використанні у виразах:

$$x = x + 1 ; ++x ; x++.$$

П



```
main.cpp X
1  #include <iostream>
2  using namespace std;
3  int main()
4  {int x, y;
5   x=5;y=6;
6   cout<<"x="<<x<<" y="<<y<<"\n";
7   cout<<"x="<<x++<<" y="<<++y<<"\n";
8
9   return 0;
10 }
11
```



```
G:\ "d:\CPP\C EDUCAT
x=5 y=6
x=5 y=7
```

Різниця в використанні префіксної `++ x` і постфіксної `x ++` форм полягає в наступному:

`x ++` - значення змінної `x` спочатку використовується у виразі, і лише потім змінна збільшується на одиницю;

`++ x` - змінна `x` спочатку збільшується на одиницю, а потім її значення використовується у виразі.

Старшинство арифметичних операцій наступне:

`++`, `--`

`-` (унарний мінус)

`*`, `/`, `%`

`+`, `-`

Операції, однакові по старшинству, виконуються в порядку зліва направо.

Звичайно ж, для того щоб змінити порядок операцій, можуть використовуватися дужки.

ОПЕРАЦІЇ ВІДНОШЕННЯ ТА ЛОГІЧНІ ОПЕРАЦІЇ

Операції відношення використовуються для порівняння. Повний список цих операцій в мові C наступний:

< Менше,
<= Менше або дорівнює,
> Більше,
> = Більше або дорівнює,
== Дорівнює,
!= Не дорівнює.

У мові C є також три логічні операції:

&& і (AND),
|| або (OR),
! ні (NOT).

ПРОГРАМУВАННЯ

Операції відношення використовуються в умовних виразах, або, коротше, умовах. Приклади умовних виразів:

$a < 0$, $101 > 105$, $'a' == 'A'$, $'a' != 'A'$

Кожен умовний вираз перевіряється: істинний він або хибний.

Точніше слід сказати, що кожен умовний вираз приймає значення "істинно" ("true") або "хибно" ("false").

Результатом логічного виразу є цілочисельне арифметичне значення. У мові C "істинно" - це ненульова величина, «хибно» - це нуль. У більшості випадків як ненульове значення "true" використовується одиниця.

ЛОГІЧНІ ОПЕРАЦІЇ

X	Y	X AND Y	X OR Y	NOT X
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

Старшинство логічних операцій і операцій відношення наступне:

!

> < >= <=

&&

||

У логічних виразах, як і у всіх інших, можливо використовувати круглі дужки, які мають найвищий пріоритет. Крім того, вони дозволяють зробити логічні вирази більш зрозумілими і зручними для читання. Умовні та логічні вирази використовуються в керуючих операторах, таких, як *if*, *for* і інших.

Особливість логічних операцій && і || полягає в тому, що якщо при обчисленні результату операції (вираз1) && (вираз2) - значення лівого операнда (вираз1) буде нульовим, то значення другого операнда на результат операції не матиме ніякого впливу. У цьому випадку другий операнд не буде підраховуватись. А отже, надії на те, що при обчисленні другого операнда може відбутися збільшення будь-якої змінної завдяки операції ++, не виправдаються. Те ж саме стосується операції ||.

ОПЕРАЦІЯ ПРИСВОЮВАННЯ

в мові C позначається просто знаком $=$.

Також є додаткові операції присвоювання $+=$, $-=$, $/=$, $*=$ та $\%=$.

Замість виразу $n = n + 5$ можна використовувати вираз $n += 5$.

Тут $+=$ адитивна операція присвоювання, в результаті виконання якої величину, що стоїть праворуч, буде додано до величини змінної зліва. аналогічно

$m -= 20$ те ж саме, що і $m = m - 20$;

$m *= 20$ те ж саме, що і $m = m * 20$;

$m /= 10$ те ж саме, що і $m = m / 10$;

$m \% = 10$ те ж саме, що і $m = m \% 10$.

Ці операції мають той же пріоритет, що і операція присвоювання $=$, тобто нижче, ніж пріоритет арифметичних операцій. Операція $x += 5$ виконується швидше, ніж операція $x = x + 5$.

ОПЕРАЦІЇ $()$ та $[]$

У мові C круглі і квадратні дужки також розглядаються як операції. Ці операції мають найвищий пріоритет.

ОПЕРАЦІЯ УМОВА ? :

Єдина, що має три операнда. Ця операція має вигляд:

$(\text{Вираз1})? (\text{Вираз2}) : (\text{вираз3})$

Обчислюється вираз (вираз1). Якщо він має нульове значення, то обчислюється вираз (вираз2). Результатом операції буде значення виразу (вираз2). Якщо значення виразу (вираз2) дорівнює нулю, то обчислюється вираз (вираз3) і його значення буде результатом операції. У будь-якому випадку обчислюється тільки один з виразів: (вираз2) і (вираз3).

Наприклад, таку операцію зручно застосовувати для знаходження найбільшого з двох чисел x і y :

$\text{max} = (x > y)? x : y;$

Якщо другий і третій операнди є величинами типу `Ivalue`, тобто можуть стояти в лівій частині операції присвоювання, то і результат даної операції є величиною типу `Ivalue`.

ОПЕРАЦІЯ SIZEOF

Операція `sizeof` має дві форми: `sizeof (тип)` або `sizeof (вираз)`. Результатом цієї операції є цілочисельне значення величини типу або виразу в байтах. При використанні другої форми значення виразу не обчислюється, а лише визначається його тип. Приклади використання: `sizeof (short int)` або `sizeof (x)`. Як тип в операції `sizeof` заборонено використовувати тип `void`.

Далі буде показано, що деякі знаки операцій мають кілька смислових значень.

РОБОТА З ДВОВИМІРНИМИ МАСИВАМИ

При описі **двовимірного** масиву оголошення має наступний вид:

тип <ім'я масиву> [розмір 1][розмір2];

Розмір масиву може задаватися константою.

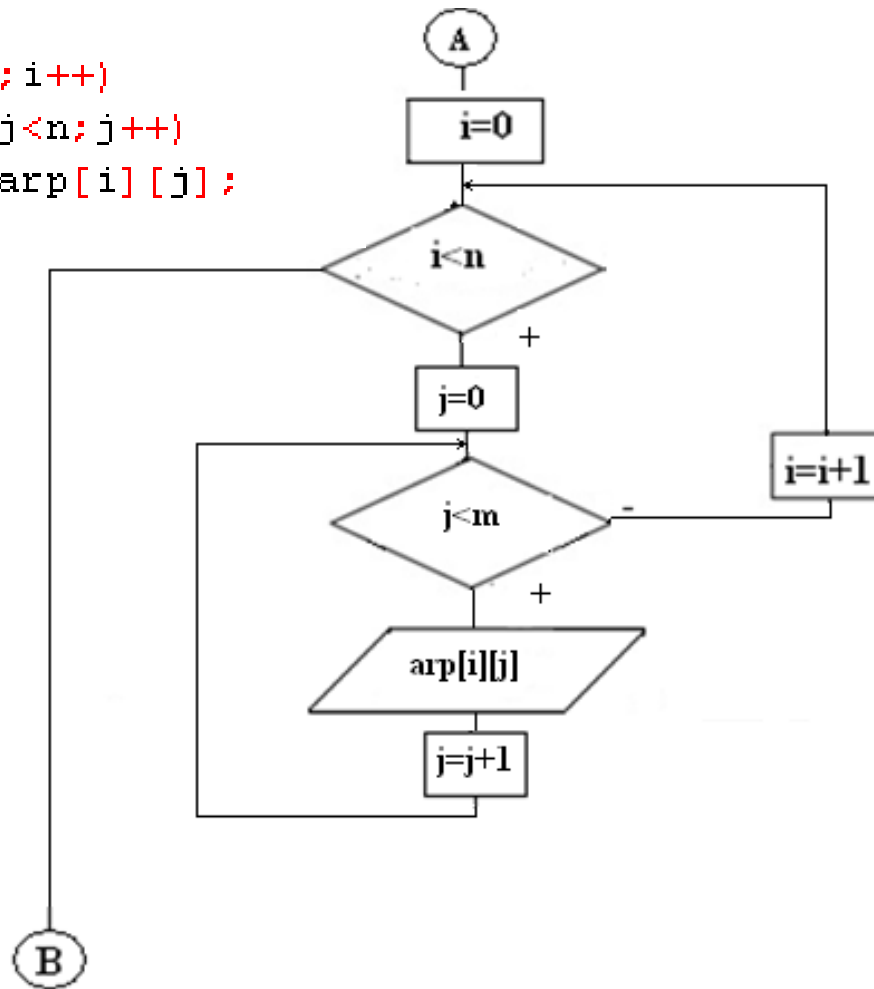
Наприклад

```
double ar1[20][30];
```

```
int ar2 [4][4];
```

Приклад. Введення з клавіатури двовимірного масиву.

```
for (i=0; i<n; i++)  
  for (j=0; j<n; j++)  
    cin>>arp[i][j];
```



Приклад. Визначення кількості нулів у двовимірному масиві

```
main.cpp X
1  #include <iostream>
2  using namespace std;
3  int main()
4  { int ar[4][2], i,j;
5    for (i=0;i<4;i++)
6      for (j=0;j<2;j++)
7        cin>>ar[i][j];
8    int zero_number=0;
9    for (i=0;i<4;i++)
10     for (j=0;j<2;j++)
11       if(ar[i][j]==0) zero_number++;
12
13     if(zero_number==0) cout<<"Number of zeros is zero"; else
14       cout<<zero_number;
15
16
17
18     return 0;
19   }
20
```

ДЯКУЮ ЗА УВАГУ!