Лекція 8

Операції мов програмування С, С++. Арифметичні операції.

Операції – це дії над даними. Дані, які беруть участь в операції, часто називають операндами. Як операнда може виступати константа, змінна або виклик будь-якої функції. Для кожної операції використовується відповідний їй знак операції, що складається з одного або декількох символів. В результаті виконання операцій завжди виходить якесь значення, що представляє собою результат виконання операції.

У мові С ++ існує велика кількість різноманітних операції. Їх можна класифікувати за різними ознаками, наприклад: за кількістю операндів, за призначенням, або якось ще.

Залежно від кількості операндів в С ++ є одномісні, двомісні і одна тримісна операція.

За призначенням операції можна згрупувати таким чином: арифметичні, операції порівняння, логічні, побітові, спеціальні.

Розглянемо тепер основні операції, а решта (спеціального характеру) будемо розбирати пізніше, коли це стане необхідно. В іншому, в таблиці нижче приведені всі знаки операції, які є в С ++, і можна відразу отримати про них певне уявлення. Операції в таблиці розбиті на групи відповідно до їх рангами.

За винятком операцій "[]", "()" і "?:", Все знаки операцій розпізнаються компілятором як окремі лексеми. Залежно від контексту одна і та ж лексема може позначати різні операції, тобто один і той же знак операції може вживатися в різних виразах і по-різному інтерпретуватися в залежності від контексту. Наприклад, бінарна операція & – це поразрядное кон'юнкція, а унарна операція & - це операція отримання адреси.

Операції рангу 1 мають найвищий пріоритет. Операції одного рангу мають однаковий пріоритет, і якщо їх в вираженні кілька, то вони виконуються відповідно до правилом асоціативності або зліва направо (→), або справа наліво (←). Якщо один і той же знак операції наведено в таблиці двічі (наприклад, знак *), то перша поява (з меншим за номером, тобто старшим за пріоритетом) відповідає унарній операції, а друга – бінарній.

Ранг	Операції	Асоциативність
1	:: . -> ( ) [ ]	→
2	! ~ + - ++ -- & * (тип) sizeof new delete тип ( ) typeid dynamic_cast static_cast reinterpret_cast const_cast	←
3	.* ->*	→
4	* / % (мультиплікативні)	→
5	+ – (адитивні)	→
6	<< >> (зсув)	→
7	< <= >= > (порівняння)	→
8	== != (порівняння)	→
9	& (поразрядна кон’юнкція І)	→

10	^ (поразрядне виключаюче АБО)	→
11	| (поразрядна диз'юнкція АБО)	→
12	&& (логічна кон'юнкція І)	→
13	|| (логічна диз'юнкція АБО)	→
14	? : (умовна операція)	←
15	= *= /= %= += -= &= ^= |= <<= >>= операція присвоювання	←
16	throw	
17	, (операція кома)	→

Арифметичні операції

Це найбільш часто вживані операції. Їх сенс близький до того, яким він відомий з курсу математики. Отже, перерахуємо їх:

Знак операції	Призначення	Приклад використання	Результат
+	додавання	**2+5**	**7**
-	віднімання	**4-1**	**3**
*	множення	**3*5**	**15**
/	ділення (звичайне)	**2.4/2**	**1.2**
/	цілочисельне ділення	**5/2**	**2**
%	Очислення залишку при цілочисельному діленні	**5%2**	**1**

Пріоритет операцій додавання і віднімання нижче, ніж множення, ділення і обчислення залишку. Для зміни порядку обчислення використовують круглі дужки, наприклад для множення на 2 суми двох чисел A і B можна написати: 2 * (A + B)

Як видно з отриманих результатів, в C ++ один знак / означає дві різні операції. Якщо один або обидва операнда – дійсні, то виконується традиційне ділення, якщо обидва операнди – цілі, то виконується ділення без остачі і результат буде цілого типу. Використання цієї операції вимагає підвищеної уважності, наприклад, якщо запрограмувати обчислення математичного виразу $\frac{1}{3}\sin(x)$ як 1/3 * sin (2 * X) то результат незалежно від значення X завжди буде дорівнює нулю, так як вираз 1/3 означає ділення без остачі. Для вирішення проблеми достатньо один з операндів зробити дійсним:

1.0 / 3 * sin (2 * X)

Операція обчислення залишку (%) може бути застосована тільки для цілочисельних операндів.

Зміна знака. Унарна операція «-» означає зміну знака. Як видно із загальної таблиці всіх операцій, вона має дуже високий пріоритет – вище,

ніж, наприклад, у операції множення. Тому в виразі -A * B спочатку виконується зміна знака для A, а потім множення -A на B.

Для парності є і операція унарний плюс, тобто можна написати + A.

Операція авто збільшення та автозменшення (інкремент та декремент)

Інкремент **A++** анологічний до дії **A+1**

Декремент **A--** анологічний до дії **A-1**

Для операцій допустимі дві форми запису:

- префіксна – наприклад, ++ A, --A
- постфіксна – наприклад, A ++, A--.

При префіксній формі запису робиться збільшення (зменшення) змінної на 1 і потім використовується нове значення цієї змінної.

У постфіксній формі запису також змінна збільшується (зменшується) на 1, але в поточному вираженні використовується старе значення змінної.

Розглянемо це на прикладах. Застосуємо постфіксну форму запису. результат:

Операції	*Значення змінної A*	*Значення змінної B*
A=1	**1**	
B=A++ **B=A--**	**2** **0**	**1** **1**

Префіксная форма запису дає такий результат:

Операції	*Значення змінної A*	*Значення змінної B*
A=1	**1**	
B=++A **B=--A**	**2** **0**	**2** **0**

Найчастіше операції використовують в операторах циклу.

З метою виконання математичних обчислень та задання складних математичних виразів розглянемо деякі стандартні математичні функції.

Стандартні математичні функції в мові C описані в заготовочному файлі **cmath.**h. Отже для їх використання на початку програми необхідно підключити цей файл за допомогою директиви **#include**.

Дамо перелік основних функцій:

Функція	**Опис**	**Приклад**
abs(a)	модуль или абсолютне значення **a**	abs(-3.0)= 3.0 abs(5.0)= 5.0
sqrt(a)	корінь квадратний з **a,** причому **a** не від'ємне	sqrt(9.0)=3.0
pow(a, b)	піднесення **a** в степінь **b** (у раціональну степінь можна підносити лише додатні числа)	pow(2,3)=8
ceil(a)	округлення **a** до найменшого цілого, но не менше ніж **a**	ceil(2.3)=3.0 ceil(-2.3)=-2.0

Функція	Опис	Приклад
floor(a)	округлення **a** до найбільшого цілого, але не більше ніж **a**	floor(12.4)=12 floor(-2.9)=-3
fmod(a,b)	Залишок від ділення a/b	fmod(4.4,7.5)=4.4 fmod(7.5,4.4)=3.1
exp(a)	обчислення экспоненти $\mathbf{e^a}$	exp(0)=1
sin(a)	**a** задається у радіанах	
cos(a)		
log(a)	натуральний логарифм **a**	log(1.0)=0.0
log10(a)	десятковий логарифм **a**	log10(10)=1
asin(a)	арксинус **a**, де **-1.0 < a < 1.0**	asin(1)=1.5708

Необхідно запам'ятати те, що операнди даних функцій завжди повинні бути дійсними, тобто *a* і *b* числа з плаваючою крапкою.

Також, зауважимо, що в мові С не існує операції піднесення до степені. Ця операція реалізована у вигляді функції **pow(a, b)**. При використанні цієї функції слід пам’ятати правила виконання показникової функції a^x. Основа ***a*** є змінною дійсного типу, але у дробову степінь (вилучення кореня) можна підносити лише додатні числа.

Для піднесення числа у степінь можна використати інший варіант із застосуванням основної логарифмічної тотожності:

$$a^b = e^{b\ln(a)} \rightarrow \exp(b*\ln(a))).$$

(аргумент логарифмічної функції a>0). Для цілих або раціональних значень степені обійти цей недолік можна таким чином: обчислити логарифм від модуля числа ***a*** і в залежності від парності значення степені присвоїти результату відповідний знак.

В мові С допускається множинне присвоєння: a=b=c=d;

Якщо тип правого операнда не збігається з типом лівого, то значення праворуч перетвориться до типу лівого операнда (якщо це можливо). При множинному присвоюванні краще дотримуватись правила зазначеного вище що старшому типу можна присвоювати молодший, а не навпаки.

Ще цікавою особливість мови С є так звана комбінована операція присвоювання виду:

a on =b

де ***on*** знак однієї з бінарних операцій: + - * / % >> << & | ^ && ||.
Присвоєння ***a on = b*** еквівалентно ***a = a on b***.

Результатом операції складеного присвоювання є значення і тип лівого операнда. наприклад:

a += b; //a = a + b;
a -=b; //a = a - b;
a *= b; //a = a * b;

a /= b; //a = a / b;
a %= b; //a = a % b;