Операції та оператори

Лектор:
к.т.н., доцент кафедри прикладної математики
Рижа Ірина Андріївна

Про що ця лекція

- Розглянемо основні операції мови С, їх класифікацію та пріоритети виконання.
- Введемо поняття оператора мови програмування.
- Опишемо основні характеристики різних типів операторів у С.

Поняття операції та виразу

Вираз

– послідовність операндів, об’єднаних символами операцій.

- В ролі операндів можна використовувати константи, змінні, імена функцій, елементи агрегованих типів, а також вирази взяті у круглі дужки.
- Вираз може містити як одну так і декілька операцій і задавати виконання послідовності елементарних кроків з перетворення інформації.

Типи операцій

Операції по кількості операндів, що беруть у ній участь, поділяються на:

- **унарні** – складаються з операнда і передуючому йому знакові унарної операції:

  знак_унарної_операції операнд;

- **бінарні** – складаються з двох операндів, розділених знаком бінарної операції:

  операнд1 знак_бінарної_операції операнд2;

- **тернарні** – складаються з трьох операндів, розділених знаками тернарної операції (?) і (:):

  операнд1 ? операнд2 : операнд3;

Унарні операції у мові С

Арифметичне заперечення (-)

змінює знак операнда на протилежний.

- Операнд повинен бути цілою або дійсною величиною.
- При виконанні здійснюються звичайні арифметичні перетворення.

```
double  u = 5;
cout <<  -u; /* вивід заперечення u, тобто значення -5 */
```

Унарні операції у мові С

Операція логічного заперечення «НЕ» (!)

повертає значення 0, якщо значенням операнда істинне (не 0), і 1, якщо значення операнда хибне (0).

- Результат має тип int.
- Будь-яке ненульове значення операнда трактується як істинне.
- Операнд повинен бути цілого або дійсного типу.

```
int z = 0;
cout <<  !z;  /* вивід НЕ z, тобто значення 1 */
```

Унарні операції у мові С

Операція інверсії (~)

повертає двійкове доповнення свого операнда.

- Операнд повинен бути цілого типу.
- Здійснюється операція заміни бітових нулів на одиницю та одиниць на нуль.

```
int z = 10;
cout <<  ~z;  /* вивід доповнення до z, тобто значення -11 */
```

z	0	0	…	0	0	0	1	0	1	0	=10
~z	1	1	…	1	1	1	0	1	0	1	
	1	1	…	1	1	1	0	1	0	0	
	1	0	…	0	0	0	1	0	1	1	=-11

Унарні операції у мові С

Операція приведення типу (type)

дозволяє задати явне перетворення операнда до нового типу даних type, де type – довільний допустимий тип даних мови С.

- Результатом операції (type)операнд є значення виразу, приведеного до типу type.

```
double v = 1.55;
char b = '9';
cout << (int)v;   /* вивід цілого значення, тобто 1 */
cout << (int)b;   /* вивід цілочисельного еквіваленту коду числа 9, тобто 57 */
```

Унарні операції у мові С

Операція sizeof()

визначає розмір пам'яті в байтах, яка відповідає ідентифікатору або типові.

- Має наступний формат: sizeof(вираз).
- Виразом може бути будь-який ідентифікатор або ім'я типу (окрім void), взяте в дужки.

```
char b = '9';
cout << sizeof(b);   /* вивід розміру пам'яті в байтах, яка відповідає змінній типу
char, тобто 1 */
cout << sizeof(int);   /* вивід розміру пам'яті в байтах, яка відповідає типу int,
тобто 4 */
```

Унарні операції у мові С

Операція збільшення (++) і зменшення (--)

відповідно збільшують або зменшують значення операнда на одиницю.

- Операнд не може бути константою.
- Тип результату відповідає типу операнда.
- ***Префіксна форма запису*** – якщо знак операції стоїть перед операндом, то зміна операнда відбувається до його використання у виразі і результатом операції є збільшене або зменшене значення операнда.
- ***Постфіксна форма запису*** – якщо знак операції стоїть після операнда, то операнд спочатку використовується для обчислення виразу, а потім відбувається зміна значення операнда.
- У випадку, якщо операції збільшення і зменшення використовуються як ***самостійні оператори***, префіксна і постфіксна форми запису стають еквівалентними.

Бінарні операції

1. **Мультиплікативні**
   - множення (*);
   - ділення (/);
   - остача від ділення (%);
2. **Адитивні**
   - додавання (+);
   - віднімання (-);
3. **Операції зсуву**
   - зсув вліво (<<);
   - зсув вправо (>>);
4. **Логічні операції**
   - логічне І (&&);
   - логічне АБО (||);
5. **Операції порівняння**
   - Менше (<)
   - Менше або дорівнює (<=)
   - Більше (>)
   - Більше або дорівнює (>=)
   - Дорівнює (==)
   - Не дорівнює (!=)
6. **Порозрядні операції**
   - порозрядне І (&);
   - порозрядне АБО (|);
7. **Послідовне обчислення**
   - операція «кома» (,);

Бінарні операції

8. **Операції присвоювання**
   - присвоєння (=);
   - множення з присвоюванням (*=);
   - ділення із присвоюванням (/=);
   - залишок від ділення з присвоюванням (%=);
   - віднімання з присвоюванням (-=);
   - додавання з присвоюванням (+=);
   - зсув вліво з присвоюванням (<<=);
   - зсув вправо з присвоюванням (>>=);
   - порозрядне І з присвоюванням (&=);
   - порозрядне АБО з присвоюванням (|=).

Операція присвоєння

Операція присвоєння (=)

змінює вміст області пам'яті, на яку посилається ідентифікатор лівого операнда

операнд1 = операнд2;

- У лівій частині оператора присвоєння обов'язково повинен бути ідентифікатор змінної (не константи).
- Правий операнд може бути будь-яким виразом: змінною, константою, літералом, арифметичним чи логічним виразом.

Перетворення типів даних

Неявне перетворення типів

Якщо операнди бінарних операцій є різного типу, то перед виконанням бінарної операції компілятор автоматично перетворює типи всіх операндів до одного типу.

Правила перетворення типів у С:

1. Операнди різних типів приводяться до «старшого», тобто типу, який займає більшу пам'ять.

   bool → char → unsigned char → ... → unsigned long → float → ... → long double

2. При виконанні операції присвоєння результат перетворюється до типу змінної, що стоїть зліва від знаку операції.
   - Може виникнути перетворення старшого типу до молодшого.

Наприклад, при переході від дійсного типу до цілого відбувається заокруглення до найближчого цілого, тобто відкидання дробової частини.

Мультиплікативні операції (*, /, %)

Операція множення операнд1 * операнд2

виконує множення операндів.

Операція ділення операнд1 / операнд2

виконує ділення першого операнда на другий (не нуль).

Операція знаходження остачі від ділення операнд1 % операнд2

повертає залишок від ділення першого операнда на другий.

- Операндами операції (%) повинні бути тільки цілі числа.
- Типи операндів операцій множення і ділення можуть відрізнятися, і для них справедливі правила перетворення типів.
- Типом результату є тип операндів після перетворення.

Адитивні операції (+, -)

Операція додавання операнд1 + операнд2

виконує додавання двох операндів.

Операція віднімання операнд1 - операнд2

віднімає другий операнд із першого.

- Операнди можуть бути цілого або дійсного типів.
- Перетворення при адитивних операціях не забезпечують обробку ситуацій переповнення і втрати значимості.

Порозрядні операції (&, |, ^)

Операндами порозрядних операцій можуть бути дані будь-якого цілого типу.

Операція порозрядного логічного «І» операнд1 & операнд2

порівнює кожен біт першого операнда із відповідним бітом другого операнда.

- Якщо обидва біти рівні 1, то відповідний біт результату – одиниця, у протилежному випадку – нуль.

- *Наприклад*, нехай c = a & b, тоді і-тий розряд за значенням визначається з таблиці:

a_i	b_i	$c_i=a_i\&b_i$
0	0	0
0	1	0
1	0	0
1	1	1

Порозрядні операції (&, |, ^)

Операція порозрядного логічного «АБО» операнд1 | операнд2

порівнює кожен біт першого операнда із відповідним бітом другого операнда.

- Якщо хоча би один з бітів дорівнює 1, то відповідний біт результату – одиниця, у протилежному випадку – нуль.

- *Наприклад*, нехай c = a | b, тоді і-тий розряд за значенням визначається з таблиці:

a_i	b_i	$c_i=a_i\|b_i$
0	0	0
0	1	1
1	0	1
1	1	1

Порозрядні операції (&, |, ^)

Операція порозрядного додавання за модулем 2 операнд1 ^ операнд2

порівнює кожен біт першого операнда із відповідним бітом другого операнда.

- Якщо вони не співпадають, то відповідний біт результату – одиниця, у протилежному випадку – нуль.

- *Наприклад*, нехай c = a ^ b, тоді і-тий розряд за значенням визначається з таблиці:

a_i	b_i	$c_i=a_i \hat{} b_i$
0	0	0
0	1	1
1	0	1
1	1	0

Операції зсуву (<<, >>)

Операції порозрядного зсуву вліво операнд1 << операнд2 **або вправо** операнд1 >> операнд2 здійснюють зсув першого операнда вліво (<<) або вправо (>>) на число бітів, що задається другим операндом.

- Обидва операнди повинні бути цілочисельними величинами.
- При зсуві вліво, вільні праві біти заповнюються нулями.
- При зсуві вправо вільні ліві біти заповнюються копією знакового біта, або нулями, якщо тип першого операнда unsigned.
- Результат операції зсуву не визначений, якщо другий операнд від'ємний.

Логічні операції (&&, ||)

Логічні операції оцінюють кожен операнд з точки зору його рівності нулеві.

- У кожній логічній операції можуть брати участь операнди різних типів.
- Операнди логічних виразів обчислюються зліва направо.
- Якщо значення першого операнда достатньо, щоб визначити результат операції, то другий операнд не обчислюється.

Операція логічного «І» операнд1 && операнд2

повертає значення 0, якщо хоча б один з операндів дорівнює 0.

- Якщо значення першого операнда рівне 0, то другий операнд не обчислюється.

Операція логічного «АБО» операнд1 || операнд2

повертає значення 1, якщо хоча б один з операндів має ненульове значення.

- Якщо перший операнд має ненульове значення, то другий операнд не обчислюється.

Тернарна операція (?)

Операція умови операнд1 ? операнд2 : операнд3;

оцінює операнд1 з погляду його істинності (еквівалентності 1).

- Якщо значення першого операнда є **істинним** (1), то обчислюється операнд2 і його значення є результатом операції.
- Якщо значення першого операнда є **хибним** (0), то обчислюється операнд3 і його значення є результатом операції.

Обчислюється лише один з операндів (або операнд2*, або* операнд3*), але не обидва.*

```
max = (d <= b) ? b : d; /*змінній max присвоюється максимальне значення
                          змінних d і b*/
```

Складене присвоєння

Складені операції присвоєння

поєднують просте присвоєння з однією з бінарних операцій:

операнд1 бінарна_операція = операнд2;

яке за результатом виконання еквівалентне наступному простому присвоєнню:

операнд1 = операнд1 бінарна_операція операнд_2;

- Вираз складеного присвоювання з точки зору реалізації не зовсім еквівалентний простому присвоюванню, оскільки в останньому – операнд1 (адреса) обчислюється двічі.

```
int a = 2, b = 3;
a += b; /* еквівалентно a = a + b;*/
a -= 3; /* еквівалентно a = a - 3;*/
b %= a; /* еквівалентно b = b % a;*/
```

Пріоритети виконання операцій

Пріоритет операції

визначає порядок трактування та виконання виразу.

- Цей порядок може змінюватися при використанні круглих дужок: (вираз).
- Якщо вираз містить операції однакового пріоритету, то компілятор враховує додатково порядок виконання операцій зліва направо чи справа наліво.
- Взятий у круглі дужки вираз опрацьовується компілятором в першу чергу.
- Якщо є дужки різного рівня занурености, то спершу виконуються вирази в «найглибших» дужках.

Пріоритети виконання операцій

Пріоритет	Знак операції	Типи операції	Порядок виконання
1	( ) [ ] . ->	Вирази	Зліва направо
2	- ~ ! * & ++ -- sizeof (type)	Унарні	Справа наліво
3	* / %	Мультиплікативні	Зліва направо
4	+ -	Адитивні	Зліва направо
5	<< >>	Зсув	Зліва направо
6	< > <= >=	Відношення	Зліва направо
7	== !=	Відношення (рівність)	Зліва направо
8	&	Порозрядне «І»	Зліва направо
9	^	Порозрядне, що виключає АБО	Зліва направо
10	|	Порозрядне АБО	Зліва направо
11	&&	Логічне «І»	Зліва направо
12	||	Логічне «АБО»	Зліва направо
13	? :	Умова	Справа наліво
14	= *= /= %= += -= &= |= >>= <<= ^=	Просте і складене присвоєння	Справа наліво
15	,	Послідовне обчислення	Зліва направо

Оператори мови С

Оператори мови програмування

керують обчислювальним процесом, дозволяють здійснювати розгалуження, циклічне повторення одного чи декількох операторів, передавати керування в потрібне місце коду програми.

- Оператори в програмі можуть об’єднувати у складені оператори за допомогою фігурних дужок.
- Кожен оператор мови (крім складених операторів) закінчується символом «;».

Класифікація операторів

1. **Умовні оператори**
   - оператор умови if;
   - оператор вибору switch;
2. **Оператори циклу**
   - for, while, do while;
3. **Оператори виходу та переходу**
   - break, continue, return, goto;
4. **Інші оператори**
   - оператор «Вираз»;
   - порожній оператор.

Оператор «Вираз»

Виконання оператору «Вираз»

полягає в обчисленні значення певного виразу.

```
++ i; /*оператор «Вираз» збільшує значення змінної і на одиницю*/

a = cos(b * 5); /*оператор «Вираз» включає в себе операції присвоєння і виклику
                  функції*/

f(x,y); /*оператор «Вираз» складається з виклику функції f */
```

Порожній оператор

Порожній оператор

– команда, яка не містить жодної інструкції, але завершується символом «;».

- При виконанні цього оператора нічого не відбувається.

```
…
int i =5; ;
…
```

Складений оператор

Складений оператор

– декілька операторів і оголошень, взятих у фігурні дужки, які трактуються у мові С як один оператор:

```
{  [оголошення]
        …
        оператор; … [оператор];
}
```

- Виконання складеного оператора полягає у послідовному виконанні його операторів.
- Після складеного оператора (закритої фігурної дужки) крапка з комою НЕ ставиться.
- Перед закриваючою фігурною дужкою непорожнього тіла складеного оператора обов'язково має стояти крапка з комою.

Оператори організації розгалужень

Оператор if

– використовується для умовної передачі керування (перевіряється умова – «якщо»).

Неповна форма оператора if використовується для організації у програмі обходів:

```
if (вираз) оператор1;
оператор2;
```

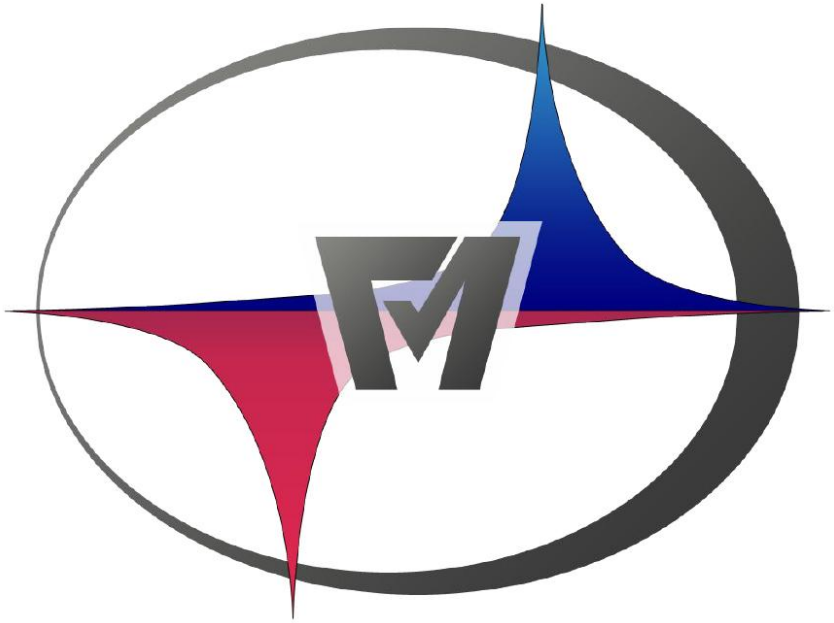
- Якщо вираз (умова) є істинним, тоді виконується оператор1, а далі оператор2.
- Якщо вираз є хибним (в тому числі набуває нульового значення), тоді одразу виконується оператор2.
- Вираз обов'язково береться у круглі дужки.
- На місці оператора1 та оператора2 можуть бути як прості, так і складені оператори.

Оператори організації розгалужень

Повна форма оператора if використовується для організації у програмі розгалужень:

```
if (вираз) оператор1;
      else оператор2;
оператор3;
```

- Якщо **вираз** (умова) є істинним, тоді виконується **оператор1**, а далі **оператор3**.
- Якщо **вираз** є хибним, тоді виконується **оператор2**, а далі **оператор3**.
- На місці кожного з операторів можуть бути як прості, так і складені оператори.
- Допускається використання вкладених операторів if, тобто оператор if може бути включений у конструкцію if або у конструкцію else іншого оператора if.
- Кожне ключове слово else відноситься до найближчого зліва if, у якому відсутній else.



Кафедра прикладної математики

http://amath.lp.edu.ua