**Основні операції**

Розглянемо способи обробки даних - для цього мову Сі має широкий набір можливостей. Основні арифметичні операції: додавання, віднімання, множення, ділення. Операції в мові Сі застосовуються для подання арифметичних дій. Наприклад, виконання операції + призводить до складання двох величин, що стоять ліворуч і праворуч від цього знака. Розглянемо операції =, +, -, \*, /. У мові Сі немає операції піднесення до степеня.

**Операція присвоєння "="**

У мові Сі знак рівності не означає "дорівнює". Він означає операцію присвоювання деякого значення. За допомогою оператора

year=2004;

змінної c ім'ям year присвоюється значення 2004, тобто елемент зліва від знака = - це ім'я змінної, а елемент праворуч - її значення. Ми називаємо символ = **операцією присвоювання**. У цій операції дія виконується справа наліво.

Можливо, відмінність між ім'ям змінної і її значенням здасться незначною? В такому випадку давайте розглянемо наступний оператор:

i=i+1;

C математичної точки зору це нісенітниця. Якщо ви додаєте одиницю до кінцевого числа, результат не може дорівнювати вихідному числу. Але як оператор присвоювання даний рядок має цілком певний сенс, який можна виразити, наприклад, такою фразою. Взяти значення змінної з ім'ям i, до нього додати 1, а потім привласнити нове значення змінної з ім'ям i.

Оператор виду

2004=year;

на мові Сі не має сенсу, оскільки 2004 - число. Ми не можемо привласнити константі якесь значення; її значенням є вона сама. Тому, пам'ятайте, що елемент, що стоїть зліва від знака =, завжди повинен бути ім'ям змінної.

***Операнд*** *- це те, над чим виконуються операції. Наприклад, можна описати процес "поїдання" картоплі як застосування операції "поїдання" до операнду "картопля".*

Операція присвєння в мові Сі видається більш цікавою, ніж в більшості інших мов. Розглянемо просту програму:

/\* таблиця результатів з шахів \*/

main( )

{

int l, m, n;

n=m=l=165;

printf("l m n\n");

printf("Рахунок першої партії %4d %8d %8d\n", l, m, n);

}

Присвоєння виконуються справа наліво: спочатку змінна l отримує значення 165, потім змінна m і нарешті n. У мові Сі є кілька інших операцій привласнення, які відрізняються від описаної операції. Їх ми розглянемо пізніше.

Уявімо ще один приклад простої програми:

/\* використання операції присвоєння \*/

main( )

{

/\* змінні number, ouzo, cost ініціалізуються конкретними значенями \*/

int number=5;

float ouzo=13.5;

int cost=31000;

printf("number ouzo cost\n");

printf("number=%d ouzo=%f cost=%d\n",

number, ouzo, cost);

}

**Операція додавання: +**

Виконання операції + призводить до складання двох величин, що стоять ліворуч і праворуч від цього знака. Наприклад, в результаті роботи оператора

printf("%d", 100 + 65);

на друк буде виведено число 165, а не вираз 100 + 65. Операнди можуть бути як змінними, так і константами. Операція + називається "бінарної", або "диадического". Ці назви відображають той факт, що вона має справу з двома операндами.

Приклад:

i=j+2;

Змінній i присвоюється значення змінної j плюс 2.

**Операція віднімання: -**

Виконання операції віднімання призводить до віднімання числа, розташованого праворуч від знака -, з числа, що стоїть зліва від цього знака. Оператор

n = 163.00 - 100.00;

присвоює змінній n значення 63.00.

**Операція зміни знаку: -**

Знак мінус використовується для вказівки або зміни алгебраїчного знака деякої величини. Наприклад, в результаті виконання послідовності операторів

teg = -15;

get = -teg;

змінній get буде присвоєно значення 15. Коли знак використовується подібним чином, дана операція називається "унарною". Таку назву вказує на те, що вона має справу тільки з одним операндом.

Приклад:

x = -x;

Операція змінює алгебраїчний знак x.

**Операция умножения: \***

Операция умножения обозначается знаком \*. При выполнении оператора

z = 3 \* x

значение переменной x умножается на 3, и результат присваивается переменной z.

**Операція ділення: /**

У мові Сі символ / вказує на операцію ділення. Величина, що стоїть зліва від цього знаку, ділиться на величину, розташовану праворуч від цього знаку. Наприклад, в результаті виконання оператора

l = 126.0 / 2.0;

змінній l буде присвоєно значення 63.0. Над даними цілого типу операція ділення проводиться не так, як над даними з плаваючою крапкою: в першому випадку результат буде цілим числом, а в другому - числом з плаваючою крапкою. У мові Сі прийнято правило, згідно з яким дрібна частина у результаті ділення цілих чисел відкидається. Ця дія називається "урізанням".

Розглянемо приклад, як виконується урізання і чим розподіл цілих чисел відрізняється від ділення чисел з плаваючою крапкою:

main( )

{

printf("ділення цілих: 5/4 це %d \n", 5/4);

printf("ділення цілих: 6/3 це %d \n", 6/3);

printf("ділення цілих: 7/4 це %d \n", 7/4);

printf("ділення чисел з плаваючою крапкою: 7./4. це %2.2f \n", 7./4.);

printf("змішане ділення: 7./4 це %2.2f \n", 7./4);

}

Останній приклад на використання змішаних типів, тобто поділ дійсного числа на ціле. Мова Сі менш суворо підходить до подібних питань і дозволяє виконувати такі операції.

|  |  |
| --- | --- |
| ! | Смешения типов следует избегать! |

Результат виконання даної програми :

ділення цілих:: 5/4 це 1

ділення цілих:: 6/3 це 2

ділення цілих:: 7/4 це 1

ділення чисел з плаваючою крапкою: 7./4. це 1.75

змішане ділення: 7./4 це 1.75

В результаті ділення цілих чисел відбувається відсікання дробової частини. Коли ми змішуємо цілі числа і числа з плаваючою крапкою, результат буде таким же, як якщо б обидва операнда були числами з плаваючою крапкою. У цьому випадку перед поділом ціле число перетворюється в число з плаваючою крапкою.

Для того щоб зрозуміти, що відбувається в тих випадках, коли в одному операторі використовується кілька операцій, розглянемо порядок виконання операцій. Цілком очевидно, що зміна порядку виконання дій може призводити до різних результатів. У мові Сі кожній операції призначається рівень старшинства. Множення і ділення мають більш високий рівень, ніж додавання і віднімання, тому вони виконуються першими. Якщо ж дві операції мають один і той же рівень старшинства, вони виконуються в тому порядку, в якому присутні в операторі.

Для більшості операцій звичайний порядок - зліва направо. Операція = є винятком з цього правила. Багато програмістів вважають за краще представляти порядок обчислень за допомогою дерева виразів. Наприклад, вираз

(a+b)\*c

буде виглядати ось так:

\*

/ \

+ c

a b

Ми можемо скласти таблицю правил описаних нами операцій. Надалі ми наведемо таблицю, де містяться правила, які стосуються усіх операцій мови Сі.

|  |  |
| --- | --- |
| Таблиця 1.1. Операції в порядку зменшення рівня старшинства | |
| **Операції** | **Порядок обчислення** |
| ( ) | Зліва направо |
| - (унарний) | Зліва направо |
| \* / | Зліва направо |
| + - (віднімання) | Зліва направо |
| = | Зліва направо |

**Додадкові операції**

У мові Сі є близько сорока операцій. Ті операції, які ми розглянули, є найбільш загальновживаними. Розглянемо ще три операції, найбільш використовувані програмістами.

**Операція ділення по модулю: %**

Ця операція використовується в целочисленной арифметиці. Її результатом є залишок від ділення цілого числа, що стоїть зліва від знака операції, на число, розташоване праворуч від неї. Наприклад, 63% 5, читається як 63 по модулю 5, має значення 3, тому що 63 = 12 \* 5 + 3.

В результаті виконання оператора

minutes=time%60;

змінній minutes буде прсвоєно значення залишку від ділення time на 60.

**Операція збільшення: ++**

Операція збільшення здійснює наступну просту дію: вона збільшує значення свого операнда на одиницю. Існують дві можливості використання даної операції: перша, коли символи ++ знаходяться зліва від змінної (операнда), - "префіксна" форма, і друга, коли символи ++ стоять праворуч від змінної, - "Постфіксна" форма. Ці дві форми зазначеної операції розрізняються між собою тільки тим, в який момент здійснюється збільшення операнда. Префіксна форма змінює значення операнда перед тим, як операнд використовується. Постфіксна форма змінює значення після того як операнд використовувався.

В прикладі,

j=i++;

змінної j спочатку присвоюється значення i, потім значення змінної i збільшується на 1.

**Операція зменшення: --**

Кожній операції збільшення відповідає деяка операція зменшення, при цьому замість символів ++ ми використовуємо -. Коли символи - знаходяться зліва від операнда - "префіксна" форма операції зменшення. Якщо символи - стоять праворуч від операнда - це "Постфіксна" форма операції зменшення.

Відповідно до прийнятого в мові Сі порядком обчислень операції збільшення та зменшення мають дуже високий рівень старшинства. Тільки круглі дужки мають більш високий пріоритет. Тому вираз a/b++ означає (a)/(b++), а не (a/b)++.

! Не застосовуйте операції збільшення або зменшення до змінної, присутньої в більш ніж одиному аргументі функції. Не застосовуйте операції збільшення або зменшення до змінної, яка входить у вираз більш ніж один раз.

### Перелік операцій мови Сі

#### Розглянемо безліч операцій мови Сі. Описані операції входять в цю множину. Мова Сі відрізняється їх великою різноманітністю. У цьому розділі буде обговорюватися їх семантика. Кожна операція характеризується рівнем пріоритету і порядком, в якому ці операції виконуються - зліва направо або справа наліво. Якщо всі операції виразу мають один і той же рівень пріоритету, то значення виразу обчислюється зліва направо відповідно до порядку виконання операцій. Всі операції з одним і тим же рівнем пріоритету мають один і той же порядок. Однак, якщо у виразі є операції з різними рівнями пріоритету, то спочатку виконуються операції з найвищим рівнем пріоритету, потім - наступного за ним пріоритету і так далі в порядку зменшення пріоритету. Операції одного рівня пріоритету виконуються в послідовності, зазначеної їх порядком.

**Операції, рівень пріоритету яких дорівнює 1**

**Операції виклику функції, індексування і вибору.** В якості операції мови Сі розглядаються також дужки у виклику функції, квадратні дужки для індексування масивів, крапка і стрілка вправо для вибору компонентів структури або об'єднання. Рівень цих операторів дорівнює 1, всі операції виконуються зліва направо.

##### Операція виклику функції: ( )

Приклад1:

fe(e1, e2,...,en);

Виклик функції fe з аргументами e1, e2, ..., en. Значення цього виразу є значення, яке повертає функція.

Приклад 2:

x = sqrt (y);

##### Операція індексування масиву: []

Синтаксис:

array [2]

Значенням виразу є третій елемент масиву.

Присвоєння значення 26 одинадцятому елементу масиву записується таким чином:

array[10]=26;

Перший елемент масиву описується виразом array[0] (більш детально про масиви описано в лекції 12).

##### Операція вибору компонентів структури або об'єднання: .

Синтаксис:

struct.element

Значенням цього виразу являється елемент element структури struct або об’єднання (див. лекцію 14). Оператор:

struct.element=1963;

присвоює значення 1963 цьому єлементу.

##### Операція вибору компонентів структури з покажчиком: ->

Синтаксис:

my\_birthday->day,

my\_birthday – показчик на структуру. Оператор:

my\_birthday->day=26;

присвоює значення 26 структурній змінній day, на яку вказує my\_birthday.

**Операції, рівень пріоритету яких дорівнює 2**

##### 

##### **Унарні операції.** Для унарних операцій потрібно тільки один операнд; ці операції або префіксні, або постфіксні. Операція sizeof має два варіанти: префіксна операція і унарна операція.

##### Операція непрямого посилання: \*

Це показчик на будь-який тип T, крім void. Тип результату T.

Використання:

\*pe

Значенням виразу є змінна, що адресується покажчиком pe.

Приклад 1:

\*ptr=c;

Приклад 2:

\*fpe;

Значенням виразу є функція, що адресується покажчиком fpe.

Приклад 3:

fpe=\*funcname;

(\*fpe)(arg1, arg2);

##### Операція отримання адреси: &

У цій операції тип операнда є змінна будь-якого типу, крім void. Тип результату - показчик на Т.

Використання:

&v

Значенням виразу є адреса змінної v.

Приклад:

myptr=&n;

##### Операція заперечення: -

Тип *операнда* - арифметичний. Тип результату: unsigned, long, double, int.

##### Операція логічного заперечення: !

Тип *операнда* – арифметичний абопоказчик. Тип результату - int. Якщо *операнд* дорівнює 0, то результат дорівнює1 і навпаки.

Приклад:

if(!good) printf("not good");

##### 

##### Операція доповнення до 1: ~

Тип *операнда* - інтегральний. Тип результату: int, long, unsigned.

Приклад:

opposite=~mask;

Доповлення до одиниці значення mask. Результат присвоюється змінній opposite.

##### Операція збільшення: ++

Тип *операнда* – арифметичний або показчик. Тип результату: int, unsigned, long, double, показчик. Значення операнда збільшується, і стає новим значенням операнда. Значення покажчика збільшується на величину зазначених вище об'єкта, значення інших операндів збільшуються на одиницю.

Операція збільшення, постфіксна: ++

Тип *операнда* – арифметичний або показчик. Тип результату: int, unsigned, long, double, показчик. Значення операнда збільшується, але повертається старе значення операнда. Значення покажчика збільшується на величину зазначеного вище об'єкта, інші операнди збільшуються на одиницю.

Використання:

iv++

Збільшення iv на 1. Значенням цього виразу є значення iv до збільшення.

Приклад:

j=i++;

Використання:

pv++;

Збільшення покажчика pv на 1, так що він буде вказувати на наступний об'єкт того ж типу. Значенням цього виразу є значення pv до збільшення.

Приклад:

\*ptr++=0;

Присвоїти начення 0 змінній, на яку вказує ptr, потім збільшити значення покажчика ptr так, щоб він вказував на наступну змінну того ж типу .

##### Операция зменшення: --

Тип *операнда* – той самий, що для ++. Тип результату - той самий, що для ++.

Використання:

--iv

Зменшення iv на 1. Значенням цього виразу є значення iv після зменшення.

Приклад:

i=--j;

##### Операція зменшення (постфіксна): --

Тип *операнда* - той самий, що для ++ (постфіксна).

Використання:

iv--

Зменшення iv на 1. Значенням цього виразу є значення iv до зменшення.

Приклад:

j=i--;

##### Операція визначення необхідної пам'яті в байтах:sizeof

Тип *операнда* – значення будь-якого типу або ім’я типу. Тип результату - unsigned. Використовується як sizeof (вираз) або sizeof (ім’я типу).

Приклад:

n=sizeof(arname)/sizeof(int);

#### Число елементів в масиві цілих чисел, яке визначається як число байт в масиві, поділене на число байт, займаних одним елементом масиву.

**Операції, рівень пріоритету яких дорівнює 3**

##### **Мультиплікативні операції.** Порядок виконання мультиплікативних операцій - зліва направо.

##### Операція множення: \*

Тип *операндів* - арифметичний. Тип результатів: int, unsigned, long, double.

Використання:

ae1\*ae2

Добуток значень ae1\*ae2.

Приклад:

z=35\*5;

##### Операція ділення: /

Тип *операндів* - арифметичний. Тип результату - int, unsigned, long, double.

Використання:

ae1/ae2

Частка від ділення ae1 на ae2.

Приклад:

i=j/7;

##### Операція отримання залишку: %

Тип *операндів* - інтегральний. Тип результату int, unsigned, long. Знак залишку машинно-залежний.

Використання:

ae1%ae2

Залишок від ділення по модулю.

Приклад:

j=i%2;

Якщо i , парне число, то j дорівнює нулю.

**Операції, рівень пріоритету яких дорівнює 4**

##### Рівень пріоритету адитивних операцій дорівнює 4. Виконуються ці операції зліва направо.

##### Операція додавання : +

Тип *операндів*:

1. арифметичний;
2. один операнд покажчик, інший - інтегрального типу; обидва операнда не можуть бути покажчиками.

Тип результату: int, unsigned, long, double, показчик. Перед складанням значення цілого операнда множиться на величину елемента даних, тип яких відповідає типу покажчика.

Використання:

ae1+ae2

Сума значеньae1 и ae2.

Приклад 1:

i=i+100;

Первісне значення i збільшує на 100.

Приклад 2:

last=arname+arsize-1;

Первісне значення last адреса останнього елемента масиву arname.

##### Операция віднімання: -

Тип *операндів*:

1. арифметичний;
2. один *операнд* - покажчик, інший - інтегрального типа;
3. *операнди* - показчики одного типу.

В першому випадку тип результату: int, unsigned, long, double. У другому випадку тип результату - покажчик. До віднімання значення цілого операнда множиться на величину елемента даних, тип яких відповідає типу покажчика. У третьому випадку тип результату - int. Результат - число об’єктів, відокремлених двома покажчиками.

**Арифметичні дії з покажчиками.** Арифметичні дії з покажчиками відрізняються від арифметичних дій зі звичайними цілими значеннями, і визначаються наступним чином. Припустимо, що i є цілим виразом, а покажчики p і q вказують на елементи типу Т. Додавання значення i та p еквівалентно додаванню числа елементів пам'яті в байтах, зайнятих i елементами типу Т.

Аналогічно визначається операція віднімання. Результат віднімання двох покажчиків типу Т є не різницею значень двох покажчиків, а числом елементів типу Т, розміщених між осередками, посилання на які забезпечуються покажчиками. Ніякі інші арифметичні дії з покажчиками не допускаються. Віднімання покажчиків має сенс тільки для покажчиків, що посилаються на елементи одного і того ж масиву, оскільки тільки в цьому випадку різниця адресів елементів масиву завжди дорівнює добутку цілого значення на величину елемента.

Використання:

ae1-ae2

Різниця значень ae1 та ae2.

Приклад 1:

i=j-100;

Використання:

pe-ie

Адреса змінної типу pe, менша на ie адресу, заданого покажчиком pe.

Приклад 2:

first=last-arsize+1;

Використання:

pe1-pe2

Число змінних типу pe в діапазоні від pe2 до pe1.

Приклад 3:

arsize=last-first;

**Операції, рівень пріоритету яких дорівнює 5**

##### Рівень пріоритету операцій зсуву дорівнює 5. Порядок виконання операцій - зліва направо.

##### Операція зсуву вліво: <<

Тип операнда - інтегральний. Тип результату - такий же, як у лівого операнда. Правий операнд перетвориться до типу int. Лівий операнд зсувається на число розрядів, що дорівнює значенню правого операнда. Звільнені розряди заповнюються нулями.

Використання:

ie1<<ie2

Двійкове представлення ie1 зсувається вліво на ie2 розрядів, звільнені розряди заповнюються нулями.

Приклад:

four=x<<2;

##### Операція зсуву вправо: >>

Тип операнда - інтегральний. Тип результату - такий же, як у лівого операнда. Правий операнд перетвориться до типу int. Лівий операнд зсувається на число розрядів, що дорівнює значенню правого операнда. Зсув буде логічним зсувом, якщо лівий операнд має тип unsigned.

Використання:

ie1>>ie2

Двійкове представлення ie1 зсувається вправо на ie2 розрядів. Зсув вправо може бути арифметичним (тобто що звільнені зліва розряди заповнюються значеннями знакового розряду) або логічним в залежності від реалізації, однак гарантується, що зрушення вправо цілих чисел без знаку буде логічним і звільнені зліва розряди будуть заповнюватися нулями.

Приклад:

x=x>>3;

**Операції, рівень пріоритету яких дорівнює 6**

##### Рівень пріоритету операцій відносини дорівнює 6. Виконуються ці операції зліва направо.

##### Операція менше: <

Тип операндів - арифметичний або покажчик. Тип результату - int.

Використання:

ae1<ae2

Істина, якщо ae1 менше, ніж ae2.

Приклад:

if(x<0) printf("negative");

##### Операція більше: >

Тип операндів - арифметичний або покажчик. Тип результату - int.

Використання:

ae1>ae2

Істина, якщо ae1 більше, ніж ae2.

Приклад:

if(x>0) printf("positive");

##### Операція менше або дорівнює: <=

Тип операндів - арифметичний або покажчик. Тип результату - int.

Використання:

ae1<=ae2

Істина, якщо ae1 менше або дорівнює ae2.

##### Операція більше чи дорівнює: >=

Тип операндів - арифметичний або покажчик. Тип результату - int.

Використання:

ae1>=ae2

Істина, якщо ae1 більше або дорівнює ae2.

**Операції, рівень пріоритету яких дорівнює 7**

##### Рівень пріоритетів операцій рівності / нерівності дорівнює 7, виконуються вони зліва направо.

##### Операція рівність ==

Тип операндів - арифметичний або покажчик. Тип результату - int.

Використання:

ie1==ie2

Істина, якщо ie1дорівнює ie2, інакше - брехня.

##### Операція нерівність: !=

Тип операндів - арифметичний або покажчик. Тип результату - int.

Єдиним цілим значенням, з яким можна порівнювати покажчики, є нульове значення!

Використання:

ie1!=ie2

Істина, якщо ie1 не дорівнює ie2.

Приклад:

while (i!=0)

i=func;

**Операція, рівень пріоритету якої дорівнює 8**

##### Рівень пріоритету операції порозрядне І рівний 8. Виконуються такі операції зліва направо.

##### Операція порозрядне І: &

Тип операндів - інтегральний. Тип результату:int, long, unsigned.

Використання:

ie1 & ie2

Побітова операція І двійкових представлень ie1 та ie2. Значення виразу містить 1 у всіх розрядах, в яких і ie1 і ie2 містять 1, і 0 у всіх інших розрядах.

Приклад:

flag=((x&mask)!=0);

**Операція, рівень пріоритету якої дорівнює 9**

##### Рівень пріоритету операції порозрядно виключає АБО дорівнює 9. Порядок виконання таких операцій зліва направо.

##### Операція порозрядне який виключає АБО: ^

Тип операндів - інтегральний. Тип результату:int, long, unsigned.

Використання:

ie1^ie2

Побітова операція виключає АБО довічних уявлень ie1 та ie2. Значення виразу містить 1 у всіх розрядах, в яких і ie1 і ie2 містять 1, і 0 у всіх інших розрядах

Приклад:

diffbits=x^y;

#### Операція, рівень пріоритету якої дорівнює 10

##### 

##### Рівень пріоритету операції порозрядно виключає АБО дорівнює 9. Порядок виконання таких операцій зліва направо.

##### Операція порозрядне що включає АБО: |

Тип операндів - інтегральний. Тип результату:int, long, unsigned.

Використання:

ie1 | ie2

Побітова операція виключає АБО довічних уявлень ie1 та ie2. Значення виразу містить 1 у всіх розрядах, в яких і ie1 і ie2 містять 1, і 0 у всіх інших розрядах

Приклад:

attrsum=attr1 | attr2;

**Операція, рівень пріоритету якої дорівнює 11**

##### Рівень пріоритету логічної (умовної) операції та дорівнює 11. Виконується операція зліва направо.

##### Операція логічне І: &&

Тип *операндів* – арифметичний або покажчик. Тип результату - int. Якщо перший операнд дорівнює 0, то результат дорівнює 0. В іншому випадку результат буде дорівнює 1, якщо другий операнд НЕ дорівнює 0, і дорівнює 0, якщо другий операнд дорівнює 0 (якщо перший операнд дорівнює 0, то другий операнд не вираховується).

Використання:

e1&&e2

Логічна операція І значень e1 і e2. Спочатку перевіряється значення e1; значення e2 перевіряється тільки в тому випадку, якщо значення e1 - Істина. Значенням виразу є Істина, якщо значення e1 і e2 - Істина.

Приклад:

if(p!=NULL&&\*p>7) n++;

#### Якщо p - ненульовий покажчик і значення змінної, на яку вказує p, більше, ніж 7, то в цьому випадку n збільшується на 1. Зверніть увагу, що якщо значення покажчика p одно NULL (0), то вираз \* p не має сенсу.

**Операція, рівень пріоритету якої дорівнює 12**

##### Рівень пріоритету операції дорівнює 12, виконуються такі операції зліва направо.

##### Операція логічне АБО: ||

Тип операндів - арифметичний або покажчик. Тип результату int. Якщо хоча б один операнд НЕ дорівнює 0, результат дорівнює 1; інакше результат буде дорівнює 0.

Використання:

e1 || e2

Логічна операція АБО значень e1 і e2. Спочатку перевіряється значення e1; значення e2 перевіряється тільки в тому випадку, якщо значення e1 - брехня. Значенням виразу є Істина, якщо істинно будь-яке значення e1 або e2.

Приклад 1:

if(x<A || x>B) printf("out of range");

**Операція, рівень пріоритету якої дорівнює 13**

##### Рівень пріоритету операції дорівнює 13, виконуються такі операції зліва направо.

##### Операція умовний оператор: ?

Тип операндів - арифметичні; другий і третій операнди можуть бути покажчиками, структурами, об'єднаннями. Тип результату:int, long, unsigned, double, покажчик, структура або об'єднання. Другий і третій операнди перетворюються до одного і того ж типу.

Умовний оператор є єдиним оператором, для якого необхідні три операнда; використовується він у такий спосіб:

a ? b : c

де a, b, c - вирази. Якщо a недорівнює 0, то результат виразу a ? b : c дорівнює b ; інакше результат дорівнює c. З двох останніх операндів обчислюється тільки один.

Використання:

ae?e1:e2

або

pe?e1:e2 (где pe - указатель)

Якщо істинно ae або pe, то виконується e1; інакше виконується e2. Значення цього виразу є значення виразу e1 або e2.

Приклад:

abs=(i<=0)?-i:i;

**Операція, рівень пріоритету якої дорівнює 14**

##### Рівень пріоритету операції дорівнює 14, виконуються такі операції справа наліво.

##### Операція просте присвоєння "="

Тип операндів: арифметичні, покажчики, об'єднання або структури. Тип результату: якщо обидва операнда мають арифметичний тип, то значення правого операнда перетвориться до типу лівого операнда.

Використання:

v=e

Присвоєння значення e змінній v.

Приклад:

x=y;

##### Операція складне присвоє "="

Позначимо <знак> один із знаків: +, -, \*, /, %, >>, <<, &, ^, |.

В результаті виконання операції присвоєння

v=e

де v - змінна, а e - вираз, значення виразу стає новим значенням змінної v.

##### Операція присвоєння

v <знак>= e

приблизно еквівалентна оператору присвоювання

v = v<знак> e

Приклад 1:

y+=2; /\* збільшення змінної y на 2 \*/

p+=n;

x-=3;

ptr-=2;

timesx\*=x;

x/=2;

x%=10;

x>>=4;

x<<=1;

remitems&=mask;

control^=seton;

additems |=mask;

Типи операндів і результату складного оператора присвоювання можна визначити на підставі цієї еквівалентності. Однак наведений еквівалент для складного оператора присвоювання не зовсім точний. У вираженні

v <знак>= e

операнд v обчислюється тільки один раз, в той час як в вираженні

v = v <знак> e

цей операнд обчислюється двічі. Ця різниця проявляється в побічних діях ї, пов'язані з обчисленням операнда v, наприклад, при зміні значення будь-якої змінної. Розглянемо це на прикладах:

a[i++]\* = n;

При виконанні обчислення лівого операнда дає побічний ефект - збільшення значення змінної i. Отже, це привласнення не еквівалентне присвоюванню

a[i++] = a[i++]\*n;

Еквівалентом першої операції присвоювання може служити послідовність операцій

a[i]=a[i]\*n;

i=i+1;

а еквівалентом другій - послідовність операцій

a[i]=a[i+1]\*n;

i=i+2;

або послідовність операцій

a[i+1]=a[i]\*n;

i=i+2;

#### в залежності від того, яка частина операції присвоювання обчислюється раніше - ліва або права. Порядок таких обчислень не визначений.

**Операція, рівень пріоритету якої дорівнює 13**

##### Рівень пріоритету операції дорівнює 13, виконуються такі операції зліва направо.

##### Операція умовний оператор: ?

Тип операндів - арифметичні; другий і третій операнди можуть бути покажчиками, структурами, об'єднаннями. Тип результату: int, long, unsigned, double, покажчик, структура або об'єднання. Другий і третій операнди перетворюються до одного і того ж типу.

Умовний оператор є єдиним оператором, для якого необхідні три операнда; використовується він у такий спосіб:

a ? b : c

де a, b, c -вирази. Якщо a не дорівнює 0, то результат виразу a? b: c дорівнює b; інакше результат дорівнює c. З двох останніх операндів обчислюється тільки один.

Використання:

ae?e1:e2

або

pe?e1:e2 (де pe - покажчик)

Якщо істинно ae або pe, то виконується e1; інакше виконується e2. Значення цього виразу є значення виразу e1 або e2.

Приклад:

abs=(i<=0)?-i:i;

**Операція, рівень пріоритету якої дорівнює 14**

Рівень пріоритету операції дорівнює 14, виконуються такі операції справа наліво.

##### Операція просте присвоєння "="

Тип операндів: арифметичні, покажчики, об'єднання або структури. Тип результату: якщо обидва операнда мають арифметичний тип, то значення правого операнда перетвориться до типу лівого операнда.

Використання:

v=e

Присвоєння значення e змінній v.

Приклад:

x=y;

##### Операція складне присвоєння "="

Позначимо <знак> один із знаків: +, -, \*, /, %, >>, <<, &, ^, |.

В результаті виконання операції присвоєння

v=e

де v - змінна, а e - вираз, значення виразу стає новим значенням змінної v.

##### Операція присвоєння

v <знак>= e

приблизно еквівалентна оператору присвоювання

v = v<знак> e

Приклад 1:

y+=2; /\* Збільшення y на 2 \*/

p+=n;

x-=3;

ptr-=2;

timesx\*=x;

x/=2;

x%=10;

x>>=4;

x<<=1;

remitems&=mask;

control^=seton;

additems |=mask;

Типи операндів і результату складного оператора присвоювання можна визначити на підставі цієї еквівалентності. Однак наведений еквівалент для складного оператора присвоювання не зовсім точний. У вираженні

v <знак>= e

*операнд* v обчислюється тільки один раз, в той час як у виразі

v = v <знак> e

цей операнд обчислюється двічі. Ця різниця проявляється в побічних діях, пов'язані з обчисленням операнда v, наприклад, при зміні значення будь-якої змінної. Розглянемо це на прикладах:

a[i++]\* = n;

При виконанні обчислення лівого операнда дає побічний ефект - збільшення значення змінної i. Отже, це привласнення не еквівалентне присвоюванню

a[i++] = a[i++]\*n;.

Еквівалентом першої операції присвоювання може служити послідовність операцій

a[i]=a[i]\*n;

i=i+1;

а еквівалентом другій - послідовність операцій

a[i]=a[i+1]\*n;

i=i+2;

або послідовність операцій

a[i+1]=a[i]\*n;

i=i+2;

в залежності від того, яка частина операції присвоювання обчислюється раніше - ліва або права**. Порядок таких обчислень не визначений.**