Вирази

Вираз являє собою об'єднання операцій і операндів. Нагадаємо, що операндом називається те, над чим виконується операція. Найпростіший вираз складається з одного операнда. Спираючись на це поняття виразу, ми можемо будувати більш складні конструкції. Наведемо кілька виразів:

100

1904 +100

a\*(c-d)

x=0

x=y++

x>3

Виразами називаються компоненти програми, складені з використанням операцій, литералов, констант, змінних (включаючи масиви, структури та об'єднання) і викликів функцій. Порядок обчислення виразів визначено лише вимогами відповідності семантиці операторів і дотримання правил пріоритету і порядку виконання операцій. При виконанні цих вимог компілятор вільний у виборі порядку обчислення виразу, навіть якщо обчислення підвиражень може призвести до побічних ефектів.

На відміну від більшості інших мов, в мові Сі для завдання певного порядку обчислення виразу недостатньо тільки відповідної розстановки дужок, так як компілятор може довільно змінювати порядок вираження, що включають асоціативні і комутативні оператори (\*, +, |, ^) навіть при наявності дужок. Для завдання бажаного порядку виконання виразу потрібно використовувати додаткові привласнення, якщо потрібно, з використанням тимчасових змінних.

Необхідно з обережністю використовувати вирази, при обчисленні яких можливі побічні результати, так як результати обчислення таких виразів часто проявляються не відразу і, крім того, залежать від використовуваного компілятора. Наприклад, в результаті обчислення операторів присвоювання

j=3;

i=(k=j+1)+(j=5);

значення змінної i дорівнюватиме 9 або 11 в залежності від того, який підвираз другої операції буде обчислено першим. Таким чином, з використанням різних компіляторів можна отримати різні результати.

Найпрстіші вирази

Найпростішими виразами називаються вирази, сформовані з використанням констант типів int, char і enum, операції sizeof, унарних операторів - і ~, бінарних операторів + \* /% & | ^ << >> = =! = <> <=> = і тернарної операції?:.

Найпростіші вирази використовуються в операторі switch, в ініціалізаторах кордонів масивів і в операторі препроцесора #if.

Логічні операції ||, && є умовними логічними операціями,так як другий операнд обчислюється тільки при необхідності. В інших мовах програмування, наприклад, в мові Паскаль, в логічних операторах завжди обчислюються значення обох операндів, навіть якщо результат може бути визначений обчисленням одного операнда.

Важливою властивістю мови Сі є те, що кожен вираз в Сі має значення. Наведемо кілька виразів і їх значення:

|  |  |
| --- | --- |
| **Вираз** | **Значення** |
| -14+16 | 2 |
| a=3+8 | 11 |
| 5>3 | 1 |
| 14<3 | 0 |
| 6+(c=3+8) | 17 |

Оператори

Будь-якеий вираз може бути перетворено в оператор додаванням до нього крапки з комою. Запис виду

вираз;

є оператором. Значення виразу ігнорується. Дія такого оператора полягає у створенні побічного ефекту обчисленням значення виразу.

Оператори служать основними будівельними блоками програми. Програма складається з послідовності операторів з додаванням невеликої кількості знаків пунктуації. Оператор є закінченою інструкцією для компіляторів. "Точка з комою" є вказівкою на наявність оператора. Тому

line = 5

- це всього лише вираз, яке може бути частиною великого виразу, але

line = 5;

є оператором.

Вираз

5+5

не є закінченою інструкцією, а служить вказівкою комп'ютера скласти 5 і 5, але не говорить, що робити з результатом.

dog=5+5;

служить вказівкою компілятору, а потім комп'ютеру, помістити результат 10 в комірку пам'яті, позначену ім'ям dog. Після занесення в пам'ять числа 10 комп'ютер може приступити до виконання наступних дій.

Складені оператори

Складений оператор являє собою два або більше операторів, об'єднаних за допомогою фігурних дужок; він називається також блоком. Щоб бути вільними в обговоренні складених операторів, розглянемо один з операторів циклу мови Сі і використання в Сі функцій printf () і scanf ().

Оператор циклу while

Нехай нам потрібно якимось чином змусити комп'ютер виконувати повторювані обчислення. Мова Сі пропонує кілька способів реалізації повторюваних обчислень. Зараз ми коротко обговоримо один з них. Даний спосіб називається "циклом з передумовою while". Цикл while працює наступним чином. Коли програма в процесі виконання вперше досягає оператора while, здійснюється перевірка істинності умови, укладеного в круглі дужки. Потім йде тіло циклу, укладену в фігурні дужки. У тілі циклу переобчислюють змінна, яка аналізується в умови, там де зустрічається закриває фігурна дужка (кінець тіла циклу while), управління передається на оператор while. Якщо умова не виконується, то управління передається за тіло циклу, тобто за закриває фігурну дужку.

Всі оператори циклу мови Сі розглядаються в 8-ій лекції.

Приклад 1:

/\* Від міста А до міста В відстань дорівнює 20 км.  
Велосипедист виїхав з А і в перший день проїхав  
10 км. В наступні дні він проїжджав зі швидкістю,  
в 1,5 разів більшою, ніж в попередній день. За скільки  
днів велосипедист добереться до міста В.\*/

#include<stdio.h>

#include<conio.h>

float km=10;

int main()

{

clrscr(); /\* чистка екрану, функція береться  
з головного файлу conio.h\*/

int d;

d=1; /\* перший день, за який велосипедист  
проїхав 10 км.\*/

while(km<20) /\* поки виконана умова циклу,  
підраховуються кілометри і дні \*/

{

km+=(km\*0.5);

d++;

}

printf("велосипедист був у шляху% d днів ",d);

getch();

}

**Вивчення і використання функцій printf () і scanf ()**

Функції printf () і scanf () дають нам можливість взаємодіяти з програмою. Ми називаємо їх функціями введення-виведення. Це не єдині функції, якими ми можемо скористатися для введення і виведення даних за допомогою програм на мові Сі, але вони найбільш універсальні. Ці функції входять в опис мови Сі і вони дані в бібліотеці stdio.h. Зазвичай функції printf () і scanf () працюють багато в чому однаково - кожна використовує керуючий рядок і список аргументів. Спочатку ми розглянемо роботу функції printf (), потім scanf ().

|  |  |
| --- | --- |
| **Формат** | **Тип інформації що виводиться** |
| %d | десяткове ціле число |
| %c | один символ |
| %s | рядок символів |
| %e | експоненціальний запис |
| %f | число с плаваючою крапкою, десятковий запись |
| %g | використовується замість запису %f або%e |
| %u | десяткове ціле число без знаку |
| %o | вісімкове ціле число без знака |
| %x | шестнадцатеричное ціле число без знака |

Інструкції, що передаються функції printf (), коли ми хочемо надрукувати деяку змінну, залежать від того, якого типу ця змінна. Наприклад, при виведенні на друк цілого числа застосовується формат% d, а при виведенні символу -% c. Формати перераховані в таблиці.

Подивимося тепер, як ці форми застосовуються:

/\* друг \*/

#include <stdio.h>

#define PI 3.14159

int main( )

{

int number = 2003;

printf("Інтернет-університет інформаційних технологій був відкритий в% d році \n", number);

printf("Значення числа pi дорівнює %f.\n", PI);

}

Формат, що вказується при зверненні до функції printf (), виглядає наступним чином:

printf(керуючий рядок, аргумент1, аргумент2,\_);

аргумент 1, аргумент 2 і т.д. - це параметри що друкуються, які можуть бути змінними, константами або навіть виразами, що обчислюються перед виведенням на друк.

Керуючий рядок - рядок символів, що показує, як повинні бути надруковані параметри. Наприклад, в операторі

printf("%d студентів отримали оцінку %f.\n",

number, z);

керуючим рядком служить фраза в лапках, а number і z - аргументи або в даному випадку значення двох змінних.  
Ми бачимо, що в керуючому рядку міститься інформація двох різних типів:

1. Символи, друкуються текстуально.
2. Ідентифікатори даних, звані також **специфікаціями перетворення**.

Кожному аргументу зі списку, наступного за керуючим рядком, повинна відповідати одна специфікація перетворення. Якщо потрібно надрукувати якусь фразу, то немає необхідності використовувати специфікацію перетворення; якщо ж потрібно тільки вивести дані на друк, то можна обійтися і без використання коментаря. Тому кожен з операторів, наведених нижче, цілком прийнятний:

printf("Ця книга не дуже дорога!\n");

printf("%c%d\n",'$',cost);

Якщо потрібно надрукувати сам символ%, то компілятор прийме його за помилкову специфікацію перетворення. Виходом із ситуації є досить просте рішення - писати два символу% підряд.

Наприклад:

int i=2+3;

printf("Тільки %d%% населення здатне вчитися самостійно! \n",i);

Результат роботи програми буде виглядати наступним чином:

Тільки 5% населення здатне вчитися самостійно!

Ми можемо розширити основне визначення специфікації перетворення, помістивши модифікатори між знаком% і символом, визначаючим тип перетворення. При використанні одночасно декількох модифікаторів вони повинні бути вказані в тому порядку, в якому перераховані в таблиці.

|  |  |
| --- | --- |
| **Модифікатори** | **Значення** |
| - | Аргумент буде друкуватися з лівої позиції поля заданої ширини. Зазвичай друк аргументу закінчується в самій правій позиції поля.  Приклад:%-10 |
| Рядок цифр | Задає мінімальну ширину поля. Більше поле буде використовуватися, якщо друковане число або рядок не поміщається в вихідному полі Приклад: %4d |
| Рядок цифр | Визначає точність: для типів даних з плаваючою крапкою число друкованих цифр праворуч від десяткового дробу; для символьних рядків - максимальне число друкованих символів Приклад:  %4.2f (дві десяткові цифри для поля шириною в чотири символи) |
| l | Відповідний елемент даних має тип long, а не int Приклад: %ld |

Приклади:

int main( )

{

printf("/%d/\n",135);

printf("/%2d/\n",135);

printf("/%10d/\n",135);

printf("/%-10d/\n",135);

}

#### Перша специфікація перетворення% d не містить модифікаторів. Це так званий вибір за замовчуванням, тобто результат дії компілятора в разі, якщо ви не дали йому ніяких додаткових інструкцій. Друга специфікація перетворення -% 2d. Вона вказує, що ширина поля повинна дорівнювати 2, але, оскільки число складається з трьох цифр, поле автоматично розширюється до необхідного розміру. Наступна специфікація% 10d показує, що ширина поля дорівнює 10. Остання специфікація% -10d також вказує ширину поля, рівну 10, а знак - приводить до зрушення всього числа до лівого краю.

#### Застосування функції scanf( )

Оскільки ми будемо користуватися функцією scanf () епізодично, ми розглянемо тут тільки основні особливості її застосування. Для функції scanf () вказується керуючий рядок і наступний за ним список аргументів. Основна відмінність функцій printf( ) і scanf () полягає в особливостях даного списку.

Функція printf( ) використовує імена змінних, константи і вирази, а функція scanf () - тільки покажчики на змінні. Ми нічого не повинні знати про такі покажчики. Необхідно пам'ятати тільки два правила:

1. Якщо нам потрібно ввести деяке значення і привласнити його змінної одного з основних типів, то перед ім'ям змінній потрібно писати символ &.
2. Якщо ми хочемо ввести значення строкової змінної, використовувати символ & не потрібно.

Приклад:

int main()

{

int age;

float assets;

char fio[50];

printf("Введіть ваш вік, оклад, фамілію. \n");

scanf("%d %f", &age, &assets);

scanf("%s", fio); /\* & відсутня при вказівці  
масиву символів \*/

printf("%d $%.0f %s\n",age, assets, fio);

}

Функція scanf( використовує деякі спеціальні знаки, пропуски, символи табуляції і "новий рядок", для розбиття вхідного потоку символів на окремі поля. Вона погодить послідовність специфікацій перетворення з послідовністю полів, опускаючи згадані спеціальні знаки між ними. Винятком є *специфікація % c*, що забезпечує читання кожного наступного символу навіть в тому випадку, якщо це порожній символ.

Функція scanf () використовує той же набір символів специфікації перетворення, що і функція printf( ) .Основні відмінності функції scanf () такі:

1. Відсутня специфікація %g.
2. *Специфікації*  %f та %e еквівалентні. Обидві специфікації допускають наявність або відсутність знака рядка цифр з десятковою крапкою або без неї і поля показника ступеня.
3. Для читання цілих чисел типу short використовується специфікація %h.

Функція scanf( ) не є однією з найбільш часто використовуваних функцій мови Сі. Ми обговорюємо її через її універсальність.

**Особливості роботи з мовою Сі.** Задання фіксованої ширини полів виявляється корисним при якому інформацію стовпчиком.

Наприклад:

printf("%d %d %5d\n",val1,val2, val3);

Результат виглядає так:

11 222 3333

4 5 23

22222 3332 11111

Ці ж дані можна представити в покращеному вигляді, якщо задати чималу фіксовану ширину поля:

printf("%9d %9d %9d\n", val1,val2, val3);

Результат буде виглядати так:

11 222 3333

4 5 23

22222 3332 11111

Якщо друковане число включено в деяку фразу, то часто при його виведенні виявляється зручним задати ширину поля рівної або менше необхідної. Це дає можливість включити число в фразу без додавання зайвих прогалин.

А тепер розглянемо два приклади роботи з циклом while:

/\*квадрати чисел\*/

int main( ) /\*отримання квадратів\*/

{

int n=1;

while(n < 11) {

printf("%10d %10d\n", n, n\*n);

n=n+1;

}

}

Ця програма виводить на друк перші 10 чисел та їх квадрати.

Другий приклад.

Згідно з легендою, один правитель обіцяв нагородити вченого, який надав йому велику послугу. Вчений, коли його запитали, що б він хотів отримати в нагороду, вказав на шахову дошку і сказав: "Покладіть одне пшеничне зерно на першу клітку, два - на другу, чотири на третю, вісім на наступну і т.д." Правитель був вражений, почувши таку скромну прохання. Програма, приведена нижче, показує, в яке становище потрапив правитель!

/\* пшениця\*/

#include <stdio.h>

#define NUMBER 64 /\* число клітинок на шаховій дошці\*/

#define CROP 7E14 /\* весь урожай пшениці,

Виражений в числі зерен\*/

int main( )

{

double current, total;

int count =1;

printf("клітинка число сума зерен доля\n");

total = current = 1.0;/\*починаємо з одного зерна\*/

printf("%4d %15.2e %13.2e %12.2e\n",count,

current, total, total/CROP);

while(count < NUMBER) {

count = count + 1;

current = 2.0\*current;

/\* подвійна кількість зерен на наступній клітці \*/

total = total +current; /\* корекція суми\*/

printf ("%4d %15.2e %13.2e %12.2e\n",count,

current, total, total/CROP);

}

}

Це приклад складеного оператора. Від відкриває фігурної дужки оператора while до закритої фігурної дужки.

*Складений оператор використовується в наступних випадках:*

1. Щоб згрупувати кілька логічних пов'язаних операторів в один оператор.
2. В якості тіла функції.
3. Для обмеження видимості визначень частиною програми, тобто для локалізації дії описів.

*Складений оператор має наступну форму*

{

визначення і опис  
 оператори

}

Визначення змінних всередині складеного оператора мають більший пріоритет, ніж визначення змінних з тим же ім'ям для області дії складеного оператора. Ці змінні видимі, тобто доступні, тільки всередині складеного оператора. Глобальні змінні є видимими всередині складеного оператора лише за умови, що їх визначення не змінені локальними визначеннями.

|  |  |
| --- | --- |
| ! | ! Застосовуйте відступи від поля в рядках, щоб зробити структуру програми наочною! |

Підведемо підсумки.

Вираз складається з операцій і операндів.

Оператор служить командою компілятору. Оператори бувають простими і складеними. Прості оператори закінчуються символом "крапка з комою". Приклади:

|  |  |
| --- | --- |
| *Оператор* опису: | int name; |
| *Оператор* присвоєння: | current=12; |
| *Оператор* виклику функції: | printf("Текст\n"); |
| Керуючий *оператор*: | while (n < 111) n=n+5; |
| Пустий *оператор*: | ; |

*Складові оператори, або блоки, складаються з одного або більше операторів, укладених у фігурні дужки, які в свою чергу самі можуть бути складовими.*

Приклади:

/\* Задача № 1.

Обчислити величину доходу по вкладу. Відсоткова  
ставка (у відсотках річних), час зберігання  
(В днях) і сума задаються під час роботи програми.\*/

#include <stdio.h>

float p,t,s,d;

int main()

{

puts("Введвть суму s");

scanf("%f",&s);

puts("Введіть відсоткову ставку р");

/\* вывод строки \*/

scanf("%f",&p);

puts("Введіть час зберігання в днях t");

scanf("%f",&t);

d=p\*s\*t/36500;

printf("\n Величина доходу d=%f",d);

}

/\* Задача № 2

За заданою стороною куба А визначити його об’єм, площу  
грані і площу поверхні.\*/

#include <stdio.h>

float a,v,s,si;

int main()

{

puts("\n Введіть сторону а");

scanf("%f",&a);

v=a\*a\*a;

s=a\*a;

si=s\*6;

printf("\n об’єм v=%f",v);

printf("\n площа грані s=%f",s);

printf("\n площа поверхні si=%f",si);

}

/\* Задача № 3

Визначити реальну відстань між населеними  
пунктами. Вказано масштаб карти і відстань між  
двома точками, які зображують населені пункти.\*/

#include <stdio.h>

float m,s,si;

int main()

{

puts("Введіть масштаб карти m");

scanf("%f",&m);

puts("Введіть відстань на карті si");

scanf("%f", &si);

s=si\*m;

printf("\n реальна відстань s=%f",s);

}