Міністерство освіти і науки України

Національний університет “Львівська Політехніка”

Лабораторна робота №7А

Виконав:
Студент групи АП-11
Білий Анатолій Іванович

Прийняв:

Чайковський І.Б.

Львів – 2024

Тема роботи: Арифметичні операції та вирази мови С.

Мета роботи: Дослідження принципів створення математичних виразів при складанні програм для виконання обчислень за допомогою різних операцій мови програмування С.

Попередні відомості.

Елементарною коміркою машинної пам’яті є біт. Біт – це елемент інформації, який може приймати значення 1 або 0. Фізично це означає наявність або відсутність електричного струму в певній ділянці електричного кола. Такий спосіб представлення елементу інформації пристосований для двійкової системи числення, яка використовується в ЕОМ. Група з восьми біт утворює байт. В одному байті можна записати беззнакове ціле число від 0 до 255 (256 - восьмий степінь числа 2) або знакове від 0 до 127. Звичайно одного байту недостатньо для запису більш складних даних, тому з двох (або чотирьох) байт утворюється машинне слово - вектор бітів, який розглядається апаратною частиною ЕОМ як єдине ціле. Число бітів у слові називається довжиною слова, залежить від апаратної реалізації комп’ютера і, як правило, буває довжиною 16 або 32 біти. Пам’ять обчислювальної машини поділяється логічно на слова. Слово має довжину, достатню для роміщення в ньому команди або цілого числа.

Всі дані, якими оперує мова С, підрозділяються на типи. Кожен тип даних має свій спосіб запису в пам’ яті ЕОМ, і, отже, займає чітко визначену ділянку. Компілятор мови С вимагає попереднього визначення типів абсолютно всіх даних, які використовуються в програмі, для того, щоб визначити спосіб і місце розміщення їх у пам’яті. При не дотриманні цих вимог робота компілятора припиняється.

Коли в програмі застосовуються ідентифікатори змінних величин, перед їх використанням обов'язково повинний бути опис типу кожної змінної, наприклад:

```
char ch;
int count = 1;
```

```
char* name = "Nick";
float ft;
double fd[30];
```

Дані розрізняються числові та текстові. Згадане вище однобайтне представлення цілого числа може бути використане як за прямим призначенням, так і для ідентифікації коду символа текстових даних. Існують стандартні (ASCII, а ткож декілька альтернативних) таблиці символів, які містять 256 цілих кодів, що відповідають найбільш поширеним текстовим символам. Можна вважати, що текстові дані представляються в пам'яті посимвольно послідовністю цілих однобайтових чисел. Таким чином, в мові "Сі" визначаються дані типу «char», які мають довжину 1 байт і можуть містити беззнакове ціле число від 0 до 255 (або від 0 до 127 зі знаком) або символьний код з таблиці..

Більшість цілих чисел в залежності від своєї величини та від апаратної реалізації ЕОМ можуть мати тип:

char - 1 байт; (символьний);
int - 2 байти;
long int - 4 байти.

Беззнакові цілі представляються модифікаціями типів «unsigned».:

unsigned char;
unsigned short int;
unsigned int;
unsigned long int.

Причиною того, що в мові С існує декілька цілих типів даних, є спроба надати можливість скористатися характерними особливостями апаратного забезпечення. На багатьох комп’ютерах між різновидами основних типів існують великі розбіжності в необхідній пам’яті, часу доступу до пам'яті та часу обчислень. Знаючи ці особливості, можна вибирти найбільш ефективний тип для певної змінної.

Для представлення чисел з плаваючою точкою існують два типи:

float - займають 4 байти;
double - 8 байт.

Число з плаваючою точкою кодується як знак, мантиса і степінь. На кожну з цих частин виділяється певна кількість бітів (в залежності від апаратної реалізації). Для підвищення точності обчислень використовується тип «double». Всі константи розглядаються як числа з подвійною точністю, всі математичні операції над нецілими числами виконуються з подвійною точністю, автоматично відбувається перетворення до вищого типу операндів, якщо вони мають різні типи, після чого здійснюється перетворення до типу, що оголошений для результуючої змінної.

В мові С існує спеціальний оператор «sizeof» для знаходження розміру об'єкта або певного типу. Це дає можливість для конкретної ЕОМ шляхом звертання з'ясувати точні розміри (в байтах), що займають числа якогось типу:

sizeof(int), sizeof(char), sizeof(double).

Користувачу надається можливість здійснювати примусове, пряме перетворення типів шляхом запису ідентифікатора типу перед іменем змінної, яка перетворюється, наприклад:

```
(float) a;
(double) f;
(int) f;
float r = float(1);
```

Крім того, основні типи можна довільно комбінувати в присвоєннях та виразах. При цьому здійснюється неявне перетворення типів за певними правилами, значення перетворюються так, щоб не було втрати інформації.

Коли із запису арифметичного виразу неясна послідовність виконуваних операцій, слід застосовувати запис у дужках.

Унарні операції та операції присвоєння правоасоціативні, всі решта - лівоасоціативні. Це означає, що, наприклад:

a = b = c те саме, що a = (b = c),

a + b + c те саме, що (a + b) + c, a

*p++ це *(p++), а не (*p)++.

При діленні додатніх цілих чисел заокруглення здійснюється в сторону 0, але якщо хоч один з операндів від'ємний, то форма заокруглення машинно-залежна. Тут завжди істинне, що :

(a/b)*b + a%b дорівнює a (якщо b не дорівнює 0).

ЗАВДАННЯ

1. Здійснити виконання програми VALUES.C:

```
#include <stdio.h>
#include <conio.h>
#include<windows.h>
int main(){
SetConsoleCP(65001);
SetConsoleOutputCP(65001);
printf("Числа типу int займають %d байт.\n",sizeof(int));
printf("Числа типу char займають %d байт.\n",sizeof(char));
printf("Числа типу float займають %d байт.\n",sizeof(float));
printf("Числа типу double займають %d байт.\n",sizeof(double));
getch();
}
```

```
Числа типу int займають 4 байт.
Числа типу char займають 1 байт.
Числа типу float займають 4 байт.
Числа типу double займають 8 байт.
```

2. Створити і виконати програми дослідження властивостей арифметичних операцій із різними типами величин.

Префіксний та постфіксний інкремент ++ і декремент --

```
#include <stdio.h>
#include <conio.h>
int main()
{ int n = 1;
 printf("n=%d \n",n);
 n++;
 printf("prefix: ++n=%d\n",++n);
 printf("postfix: n++=%d\n",n++);
 printf("after-postfix: n=%d\n",n);
 n--;
 printf("prefix: --n=%d\n",--n);
 printf("postfix: n--=%d\n",n--);
 printf("after-postfix: n=%d\n",n);
 }
```

```
n=1
prefix: ++n=2
postfix: n++=2
after-postfix: n=3
prefix: --n=2
```

postfix: n--=2
after-postfix: n=1

№ п/п	Завдання 2
1	1) n+(++m) 2) m-- *n 3)
2	1) ++n*++m 2) --n -m
3	1) n---m 2) m--*n
4	1) n++*m 2) n++/m
5	1) - -m-++n 2) m*n/n++
6	1) m-++n 2) ++m*--n
7	1) m+--n 2) m++-(++n)
8	1) n++-m 2) m-- *n
9	1) ++n*++m 2) m++-n
10	1) n---m 2) m--*n
11	1) n++*m 2) n++-m

№ п/п	Завдання 2
12	1) - -m-++n 2) m*--n
13	1) m-++n 2) ++m*--n
14	1) m+--n 2) m++*++n

Завдання варіант №1

```
#include <stdio.h>
int main() {
int n = 1, m = 1, res1, res2;
res1 = ++n * ++m;
printf("res1=%d\n", res1);
res2 = m++ - n;
printf("res2=%d", res2);
return 0;
}
```

```
res1=4
res2=0
```

Виконати приклади і пояснити результати

4.1

```
#include <stdio.h>
int main()
{ int a, b=3;
float c;
c = b%2 + (a = ++b/2) + 1.1;
printf("a=%d, c=%4.1f\n",a,c); }
```

---

a=2, c= 4.1

4.2

```
#include <stdio.h>
int main()
{
int x=2,z;
float y = 2.1;
z = x++*y + y/x*3;
printf("x=%d z=%d\n",x,z);}
```

---

x=3 z=6

4.3

```
#include <stdio.h>
int main()
{
float x = 1.1, y = 0, z;
int a;
z = (a=x++)*y + 3*x;
printf("z=%4.1f\n",z);}
```

---

z= 6.3

Висновок: під час виконання цієї лабораторної роботи я дослідив принцип створення математичних виразів при виконання обчислень за допомогою різних операцій мови програмування С.