Міністерство освіти і науки України Національний університет “Львівська Політехніка”

Лабораторна робота №3а

з дисципліни «Програмування частина 2»

Виконав:

Студент групи АП-11

Іщенко Василь

Прийняв:

Чайковський І.Б.

Львів – 2024

Тема роботи: Логічні і бітові операції та вирази мови С.

Мета роботи: Дослідження властивостей операцій порівняння, логічних і бітових мови програмування С.

Попередні відомості. Операції порівняння – бінарні, причому обидва операнди повинні бути арифметичного типу, або вказівниками. Результат цілочисельний: 0 (хибність) або 1 (істинність). Тип результату int. вираз < вираз вираз > вираз вираз <= вираз вираз >= вираз Операції рівності і нерівності відносять до цієї ж групи. Важливо правильно витримувати синтаксис знаку «логічне дорівнює» - ця операція не виконує присвоювання: вираз == вираз вираз != вираз Результатом цих операцій є 0, якщо задане відношення хибне, і 1, якщо істинне. Тип результату int. Ці операції мають нижчий пріоритет, ніж операції попередньої групи, наприклад, у виразі a < b = = c < d спочатку здійснюються порівняння a < b та c < d, результати кожного з них мають значення 0 або 1, після чого операція = = дає результат 0 або 1. Для логічних операцій характерне те, що і операнди, і результат мають цілий тип і трактуються як логічні (“Так” – 1, “Ні” – 0)

1.Операції зсуву (визначені тільки для цілочисельних операндів): операнд лівий операція зсуву операнд правий << – зсув ліворуч бітового представлення лівого цілочисельного операнда на кількість розрядів, що дорівнює значенню правого цілочисельного операнда. >> – зсув праворуч бітового представлення лівого цілочисельного операнда на кількість розрядів, що дорівнює значенню правого цілочисельного операнда.

2. Доповнення (бітове НЕ): ~ операнд Це унарна операція, яка доповнює значення біту кожного розряду операнду до 1. Операнд повинен мати тип int.

3. Бітове І: вираз & вираз Результатом є бітова функція І операндів. Результат обчислюється як бітовий – для кожного розряду операндів згідно таблиці істинності операції логічне І і записується у відповідний розряд. Операція застосовується тільки до операндів типу int.

4. Виключене бітове АБО (XOR): вираз ^ вираз Результатом є бітова функція виключене бітове АБО, яка виконується для кожного розряду операндів згідно наведеної таблиці істинності. Застосовується тільки до операндів типу int. В інших мовах програмування символ ^ застосовують для виконання операції піднесення до степеня. В мові С піднесення до другого або третього степенів зручно виконувати простим перемноженням. В інших випадках для піднесення числа x до степеня y слід використовувати вбудовану функцію pow(x,y).

ЗАВДАННЯ

1. Здійснити виконання програми порівняння двох чисел:

```
#include <stdio.h>
#include <windows.h>
void main(){
 SetConsoleCP(65001);
 SetConsoleOutputCP(65001);
 float var1, var2;
 printf("Введіть перше число (var1): ");
 scanf("%f", &var1);
 printf("Введить друге число (var2): ");
 scanf("%f", &var2);
 printf("var1 > var2 дає %d\n", var1 > var2);
 printf("var1 < var2 дає %d\n", var1 < var2);
 printf("var1 == var2 дає %d\n", var1 == var2);
 printf("var1 >= var2 дає %d\n", var1 >= var2);
 printf("var1 <= var2 дає %d\n", var1 <= var2);
 printf("var1 != var2 дає %d\n", var1 != var2);
 printf("!var1 дає %d\n", !var1);
 printf("!var2 дає %d\n", !var2);
 printf("var1 || var2 дає %d\n", var1 || var2);
 printf("var1 && var2 дає %d\n", var1 && var2);
}
```

```
Введіть перше число (var1): 1
Введить друге число (var2): 1
var1 > var2 дає 0
var1 < var2 дає 0
var1 == var2 дає 1
var1 >= var2 дає 1
var1 <= var2 дає 1
var1 != var2 дає 0
!var1 дає 0
!var2 дає 0
var1 || var2 дає 1
var1 && var2 дає 1
```

2. Здійснити модифікацію та виконання програми згідно взірця, показаного нижче..

```
#include <stdio.h>
#include <windows.h>
#define TRUE "ІСТИНА"
#define FALSE "ХИБНІСТЬ"
void main(){
 SetConsoleCP(65001);
 SetConsoleOutputCP(65001);
 float var1, var2;
 printf("Введіть перше число (var1): ");
 scanf("%f", &var1);
 printf("Введіть друге число (var2): ");
 scanf("%f", &var2);
 printf("var1 > var2 це %s\n", var1 > var2 ? TRUE : FALSE);
 printf("var1 < var2 це %s\n", var1 < var2 ? TRUE : FALSE);
 printf("var1 == var2 це %s\n", var1 == var2 ? TRUE : FALSE);
 printf("var1 >= var2 це %s\n", var1 >= var2 ? TRUE : FALSE);
 printf("var1 <= var2 це %s\n", var1 <= var2 ? TRUE : FALSE);
 printf("var1 != var2 це %s\n", var1 != var2 ? TRUE : FALSE);
 printf("var1 || var2 це %s\n", var1 || var2 ? TRUE : FALSE);
 printf("var1 && var2 це %s\n", var1 && var2 ? TRUE : FALSE);
 printf("!var1 це %s\n", !var1 ? TRUE : FALSE);
 printf("!var2 це %s\n", !var2 ? TRUE : FALSE);
}
```

```
Введіть перше число (var1): 1
Введіть друге число (var2): 1
var1 > var2 це ХИБНІСТЬ
var1 < var2 це ХИБНІСТЬ
var1 == var2 це ІСТИНА
var1 >= var2 це ІСТИНА
var1 <= var2 це ІСТИНА
var1 != var2 це ХИБНІСТЬ
var1 || var2 це ІСТИНА
var1 && var2 це ІСТИНА
!var1 це ХИБНІСТЬ
!var2 це ХИБНІСТЬ
```

3. Створити програму для виконання прикладу:

```
#include <stdio.h>
void main(){
 int x,y,z;
 x=2; y=1; z=0;
 x=x && y || z;
 printf ("%d\n",x);
 printf("%d\n",x || !y && z);
}
```

1
1

4. Пояснити результати роботи програм:

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a = 0, b = 3,c;
 c = b%2 || (a >= 0) && (++b/2*a)==0;
 printf("a=%d, c=%d\n",a,c); /*a=0,c=1*/
 getch();
}
```

---

a=0, c=1

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a = 1, b = 0,c;
 c = b*2 || (a >= 0) && (++b*a)==0;
 printf("c=%d\n",c); /*c=0*/
 getch();
}
```

---

c=0

```
#include <stdio.h>
#include <conio.h>
void main()
{ int x=1, y=2, z;
 z = (x/2*7 <= 0) && (y < 0) || (y%x == 0);
 printf("z=%d\n",--z); /*z=0*/
 getch();
}
```

---

z=0

```
#include <stdio.h>
#include <conio.h>
void main()
{ int x = 1,z,b = 0,y = 2;
 z = (x++*y >= 0) || b++ || (x/y*3 == 0);
 printf("z=%d\n",z); /*z=1*/
 getch();
}
```

---

z=1

```
#include <stdio.h>
#include <conio.h>
void main()
{ int x = 1, y = 0, z = 2; int a = 0;
 z = ((a=x++)*y == 0 || a < 0 && z);
 printf("z=%d\n",z); /*z=1*/
 getch();
}
```

---

z=1

```
include <stdio.h>
#include <conio.h>
void main()
{
int x = 2,z,y = 0;
z = (x == 0) && (y=x) || (y > 0);
printf("z=%d\n",z); /*z=0*/
getch();
}
```

---

z=0

```
#include <stdio.h>
#include <conio.h>
void main()
{ int x = 0,y = 3,z;
 z = (++x > y || y-- && y > 0);
 printf("z=%d\n",z); /*z=1*/
 getch();
}
```

---

z=1

5. Виправити помилки в прикладах:

```
#include <stdio.h>
#include <conio.h>
void main(){
unsigned int x=2,y = 1, z=3,res;
 char chx = 0xAF;
 printf("%u\n",x&y|z); /*3*/
 x=y=z=2;
 printf("%u\n",x|y&z); /*2*/
 x=3; y=0; z=1;
 printf("x^y|~z=%hu\n",x^y|~z); /*65535*/
 printf("3|0^~1=%hu\n",x|y^~z); /*65535==11111111*/
 x=1;y=2;z=0;
 printf("1&2|0=%u\n",x&y|z); /*0*/
 printf("~1^2&0=%hu\n",~x^y&z); /*65534==11111110*/
 printf("2|0&1=%u\n",y|z&x); /*2*/
 printf("2++&~0|~1=%hu\n",y++&~z|~x);/*65534==11111110*/
 printf("~3|1&++0=%hu\n",~y|x&++z); /*65533==11111101*/
 x = 0xAF; printf("%X\n",x>>4); /*A*/
 chx<<=7; printf("0x=%X\n",chx); /*(FF)80 ==10000000*/
 getch();}
```