Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут телекомунікацій, радіоелектроніки та електронної техніки
кафедра «Радіоелектронні пристрої та системи»

Звіт з лабораторної роботи №7а

з дисципліни «Програмування»

Підготував:
ст. групи АП-11
Фостик В. І.

Прийняла:
Гордійчук-Бублівська О. В.

Львів – 2024

Тема:
Арифметичні операції та вирази мови С.

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

Теоретичні 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.

Завдання:
1. Здійснити виконання програми VALUES.C:

```
#include <stdio.h>
#include <conio.h>
int main()
{
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",siezof(double));
getch();
}
```

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);
}
```

3. Виконати завдання згідно варіанта і пояснити результат при n=1, m=1.
Зразок програми.

```
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;
}
```

7	1) m+--n 2) m++-(++n)

4. Виконати приклади і пояснити результати

```
#include <stdio.h>
#include <conio.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);
}
#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);
}
#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); }

Виконання роботи:

1.

```c
#include <stdio.h>

int main() {
    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));
    return 0;
}
```

```
/tmp/Bm8fZo5KAl.o
Числа типу int займають 4 байт.
Числа типу char займають 1 байт.
Числа типу float займають 4 байт.
Числа типу double займають 8 байт.


=== Code Execution Successful ===
```

2.

```c
#include <stdio.h>
int main() {
    int n = 1;
    printf("n=%d \n", n);

    // Префіксне інкрементування (++n)
    printf("prefix: ++n=%d\n", ++n);
    // Постфіксне інкрементування (n++)
    printf("postfix: n++=%d\n", n++);
    // Після постфіксного інкрементування (перевірка значення n)
    printf("after-postfix: n=%d\n", n);
    // Префіксне декрементування (--n)
    printf("prefix: --n=%d\n", --n);
    // Постфіксне декрементування (n--)
    printf("postfix: n--=%d\n", n--);
    // Після постфіксного декрементування (перевірка значення n)
    printf("after-postfix: n=%d\n", n);
    return 0;
}
```

```
/tmp/lSYg5sU4pT.o
n=1
prefix: ++n=2
postfix: n++=2
after-postfix: n=3
prefix: --n=2
postfix: n--=2
after-postfix: n=1


=== Code Execution Successful ===
```

3.

```c
#include <stdio.h>

int main() {
    int n = 1, m = 1, res1, res2;


    res1 = m + --n;

    res2 = m++ - (++n);

    printf("res1=%d\n", res1);
    printf("res2=%d\n", res2);
    return 0;
}
```

```
/tmp/aOTgME4TKf.o
res1=1
res2=0


=== Code Execution Successful ===
```

4.

```c
#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);

    return 0;
}
```

```
/tmp/ILQfOGWLiE.o
a=2, c= 4.1


=== Code Execution Successful ===
```

4.2.

```c
#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);

    return 0;
}
```

```
/tmp/qAyBxDqbVN.o
x=3 z=6


=== Code Execution Successful ===
```

4.3.

```c
#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);

    return 0;
}
```

```
/tmp/2AcLAA9yYr.o
z= 6.3


=== Code Execution Successful ===
```