

```

/* C++ PROGRAM TO DEMONSTRATE THE UNARY OPERATOR ++ OVERLOADING
 * (PREFIX) AND RETURNING NAMELESS OBJECTS */

/* NAME: SAGAR GIRI, ROLL NO. 205, SECTION : A*/
#include <iostream>
using namespace std;
class Counter
{
    private :
        unsigned int count;
    public :
        Counter() //DEFAULT CONSTRUCTOR
        {count = 0;}

        int getcount()
        {
            return count;
        }

        Counter operator++() //CALL FOR PREFIX OPERATION
        {
            ++count;
            Counter temp;
            temp.count = count;
            return (temp);
        }
};

int main()
{
    Counter c1, c2; //CREATE 2 OBJECTS
    cout << "Count one =" << c1.getcount() <<endl;
    cout << "Count two =" << c2.getcount() <<endl;
    ++c1; //INCREMENT COUNT OF OBJECT c1 BY 1
    ++c2; //INCREMENT COUNT OF OBJECT c2 BY 1

    c2 = ++c1; //INCREMENT COUNT OF OBJECT c1 BY 1 AND ASSIGN TO c2

    cout << "Count one = " << c1.getcount() <<endl;
    cout << "Count two = " << c2.getcount() <<endl;
}

```

OUTPUT:

```

Count one =0
Count two =0
Count one = 2
Count two = 2

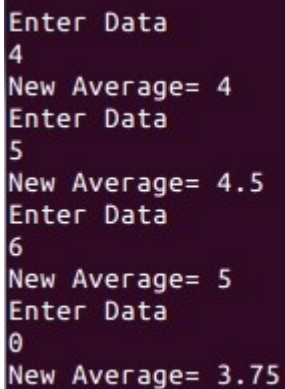
```

```
/*THIS C++ PROGRAM ILLUSTRATES THE CONCEPT OF THE STATIC LOCAL VARIABLE.  
* LIFETIME OF STATIC VARIABLE IS THROUGHOUT THE PROGRAM */
```

```
/* NAME: SAGAR GIRI, ROLL NO. 205, SECTION : A*/
```

```
#include <iostream>  
using namespace std;  
float findavg(float );  
int main()  
{  
    float avg,data=1;  
    while(data!= 0)  
    {  
        cout<<"Enter Data"<<endl;  
        cin>>data;  
        avg = findavg(data);  
        cout<<"New Average= "<<avg;  
    }  
    return 0;  
}  
float findavg(float d)  
{  
    static int count = 0; //static local integer variable  
    static float total = 0.0; //static local float variable  
    count++;  
    total = (total+d);  
    return (total/count);  
}
```

OUTPUT:



```
Enter Data  
4  
New Average= 4  
Enter Data  
5  
New Average= 4.5  
Enter Data  
6  
New Average= 5  
Enter Data  
0  
New Average= 3.75
```

```
/*C++ PROGRAM TO SHOW THE DEMONSTRATION OF OVERLOADING BINARY
* '-' OPERATOR */
```

```
/* NAME: SAGAR GIRI, ROLL NO. 205, SECTION : A*/
```

```
#include <iostream>
using namespace std;
class Coordinate
{
    private :
        int xco, yco;
    public :
        Coordinate() //DEFAULT CONSTRUCTOR
        {xco = 0; yco = 0;}

        Coordinate(int x, int y) //TWO ARGUMENT CONSTRUCTOR
        {
            xco = x;
            yco = y;
        }

        Coordinate operator-() //OVERLOADING '-' OPERATOR
        {
            return Coordinate(-xco, -yco);
        }

        void display()
        {
            cout << "(" << xco << "," << yco << ")";
        }
};

int main()
{
    Coordinate P1(5, -2), P2;
    cout <<"P1 ="; P1.display();
    cout <<endl;
    cout <<"P2 ="; P2.display();
    cout <<endl;

    P2 = P1;

    cout <<"P1 ="; P1.display();
    cout <<endl;
    cout <<"P2 ="; P2.display();
}
```

OUTPUT:

```
P1 =(5,-2)
P2 =(0,0)
P1 =(5,-2)
P2 =(5,-2)
.....
```

```

/* C++ PROGRAM TO DEMONSTRATE THE UNARY OPERATOR ++ OVERLOADING
 * (POSTFIX) AND RETURNING NAMELESS OBJECTS */

/* NAME: SAGAR GIRI, ROLL NO. 205, SECTION : A*/

#include <iostream>
using namespace std;
class Counter
{
    private :
        unsigned int count;
    public:
        Counter() //DEFAULT CONSTRUCTOR
        {count = 0;}

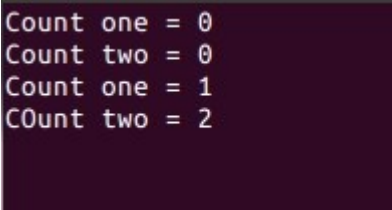
        int getcount()
        {
            return (count);
        }

        void operator ++(int) //OPERATOR CALL FOR POSTFIX
        {
            count++;
        }
};

int main()
{
    Counter c1, c2;
    cout << "Count one = " << c1.getcount() << endl;
    cout<< "Count two = " << c2.getcount() << endl;
    c1++; //INCREMENT COUNT OF C1 BY 1
    c2++; //INCREMENT COUNT OF C2 BY 1
    c2++; //INCREMENT COUNT OF C2 BY 1
    cout << "Count one = " << c1.getcount() << endl;
    cout << "C0unt two = " << c2.getcount() << endl;
}

```

OUTPUT:



```

Count one = 0
Count two = 0
Count one = 1
C0unt two = 2

```

```
/*THIS C++ PROGRAM ILLUSTRATES THE CONCEPT OF UNARY  
 * OPERATOR '++' (POSTFIX) OVERLOADING AND RETURNING NAMELESS OBJECT.  
*/
```

```
/* NAME: SAGAR GIRI, ROLL NO. 205, SECTION : A*/
```

```
#include <iostream>  
using namespace std;  
class Counter  
{  
    private:  
        int count;  
    public:  
        Counter() //DEFAULT CONSTRUCTOR  
        {count =0;}  
  
        Counter(int c) //ONE ARGUMENT CONSTRUCTOR  
        {  
            count = c;  
        }  
  
        int getcount()  
        {  
            return count;  
        }  
  
        //OPERATOR OVERLOADING FOR ++ POSTFIX OPERATION  
        Counter operator++(int)  
        {  
            return Counter(count++);  
            /*It does three functions  
            * 1. create nameless object  
            * 2. return count 2 and  
            * 3. increase count by 1 */  
        }  
};  
  
int main()  
{  
    Counter C1, C2;  
    cout << "Count one = " << C1.getcount() << endl;  
    cout << "Count two = " << C2.getcount() << endl;  
    C1++;  
    C1++;  
    C2 = C1++;  
    cout << "Count one = " << C1.getcount() << endl;  
    cout << "Count two = " << C2.getcount() <<endl;  
}
```

OUTPUT:

```
Count one = 0  
Count two = 0  
Count one = 3  
Count two = 2
```

```

/*C++ PROGRAM TO SHOW THE DEMONSTRATION OF OVERLOADING BINARY
 * "+" OPERATOR */

/* NAME: SAGAR GIRI, ROLL NO. 205, SECTION : A*/
#include <iostream>
using namespace std;
class Distance
{
    private:
        int feet; float inches;
    public:
        Distance() //DEFAULT CONSTRUCTOR
        {feet = 0; inches= 0.0;}
        Distance(int ft, float in) //TWO ARGUMENT CONSTRUCTOR
        { feet = ft; inches = in;}
        Distance operator+(Distance dd2) //OVERLOADING '+' OPERATOR
        {
            int ft;float in;
            ft = feet+dd2.feet;
            in = inches+dd2.inches;
            if(in >= 12.0)
            {
                in -= 12.0;
                ft++;
            }
            return Distance(ft,in);
        }
        void display()
        {
            cout<<feet<<"' - "<<inches<<"\""<<endl<<endl;
        }
};

int main()
{
    Distance d1(5,7.3),d2(6,9.4),d3;
    d3=d1+d2;
    cout<<"sum of ";d1.display();cout<<"and ";d2.display();

    cout<<"is: ";d3.display();
    return 0;
}

```

OUTPUT:

```

sum of 5' -7.3"
and 6' -9.4"
is: 12' -4.7"

```