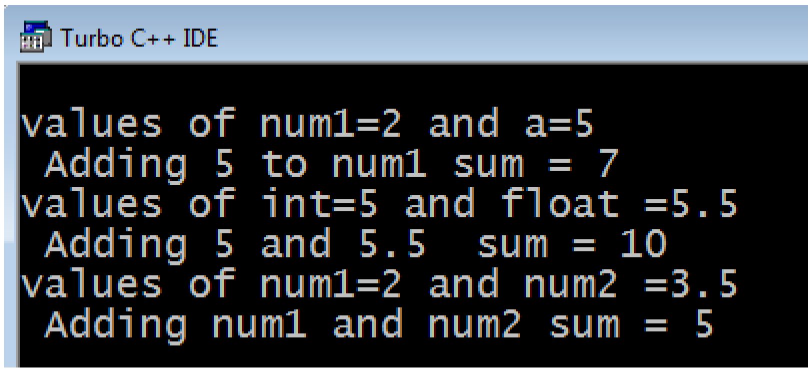
Overloading:
Overloading can of the following type
i. Function overloading
ii. Operator Overloading

Function Overloading
C++ supports function overloading which allows the same name to be given to different functions. The distinction of functions is made based on the arguments passed to the functions. The overloaded member function are selected for invoking by matching the arguments, both type and the numbers. This information is known to the compiler at compile time and therefore the compiler is able to select the appropriate function for a particular call at the compile time itself.

```
        Void  xyz (int x);
        Void xyz (int x, int y)
        Void xyz (float x, float y)
        Void xyz (float x, int y)
```

```
#include<iostream.h>
#include<conio.h>
class ex_ovrld
{
        int num1, sum;
        float num2;
 public:
 ex_ovrld ( ) {    }
  ex_ovrld(int a, float b)
  {
        num1 = a;
        num2 = b;
  }
  void add()
  {
         cout<<"\nvalues of num1="<<num1<<" and num2 ="<<num2;
         sum = num1+num2;
  }
  void add (int a)
  {
        cout<<"\nvalues of num1="<<num1<<" and a="<<a;
        sum = num1 + a;
  }
  void add (int a, float b)
  {
          cout<<"\nvalues of int="<<a<<" and float ="<<b;
          sum = a+b;
  }
  void disp();
  };
  void ex_ovrld :: disp()
  {

        cout<<" sum = "<<sum;
  }
 void main()
 {
        clrscr();
        ex_ovrld obj(2,3.5);
        obj.add(5);
        cout<<"\n Adding 5 to num1";
        obj.disp();
        obj.add(5, 5.5);
        cout<<"\n Adding 5 and 5.5 ";
        obj.disp();
        obj.add();
        cout<<"\n Adding num1 and num2";
        obj.disp();
        getch();
 }
```



```
values of num1=2 and a=5
 Adding 5 to num1 sum = 7
values of int=5 and float =5.5
 Adding 5 and 5.5  sum = 10
values of num1=2 and num2 =3.5
 Adding num1 and num2 sum = 5
```

Operator Overloading

Unary operator (unary +, unary -, postfix ++, prefix ++)

The prefix and the postfix operator overloading in c++

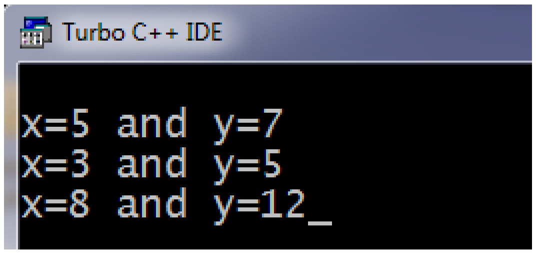
Example of Binary overloading (+, *, / etc)

```
#include<iostream.h>
#include<conio.h>

class op_ov
{
  int x, y;
  public:
  op_ov(){}
  op_ov(int a, int b)
  {
    x = a;
    y = b;
  }
  op_ov operator +(op_ov obj)
  {
        op_ov tmp;
        tmp.x = x + obj.x;
        tmp.y = y + obj.y;
  return tmp;
  }
  void disp ()
  {
  cout<< "\nx="<<x<< " and y=" << y;
  }

};
  void main()
{
                op_ov ob1(5,7), ob2(3,5), ob3;
                ob3= ob1+ob2;
                ob1.disp();
                ob2.disp();
                ob3.disp();

 }
```



```
x=5 and y=7
x=3 and y=5
x=8 and y=12_
```

Exercise:
WAP in C++ to overload the binary operator *(multiplication) and (/) division