```
#include <iostream>
using namespace std;
struct comp
{
private:
    int real;
    int img;
public:
    void setReal(int R)  {real = R;} ///setters
    void setImg(int I)  {img = I;}
    int getReal()  {return real;}  ///getters
    int getImg()  {return img;}
    ///construcor
    comp(int R,int I)
    {real =R;img =I;}

    comp(int n)
    {real =img =n;}

    comp()
    {real=img=0;}
    ///distructor
    ~comp()
     {cout<<"\nparameter destructor"<<endl;}
    ///
    void SetComp(int R, int I)
      {
        real = R;img = I;
       cout<<"\nparameter costructor"<<endl;
      }

    void SetComp(int N)
     {real = img = N;}

    void SetComp(int N, char* M)
      { real = img = N;}

    comp Addcomplex(comp C)
    {
    comp Res;
    Res.real= this -> real +C.real;
    Res.setImg(img + C.img);
    return Res;
    }
    comp subComplex(comp c)
    {
    comp Res;
    Res.setReal(getReal()-c.getReal());
    Res.setImg(getImg()-c.getImg());
    return Res;
    }
    comp PrintComplex ()
    {
        cout<<"\nResult= "<<real<<"+"<<img<<"i"<<endl;
    }
};
int main()
{
    int r1,i1,r2,i2;
    comp A,B,C;  ///comp A(3,4),B(2),C
    cout<<"Enter Real, Img Values: \n";
    cin>>r1>>i1;
    cin>>r2>>i2;
    A.SetComp(r1,i1);
    B.SetComp(r2,i2);
   // A.PrintComplex();
// B.PrintComplex();
 C = A.Addcomplex(B);
 C.PrintComplex();
 return 0;
```