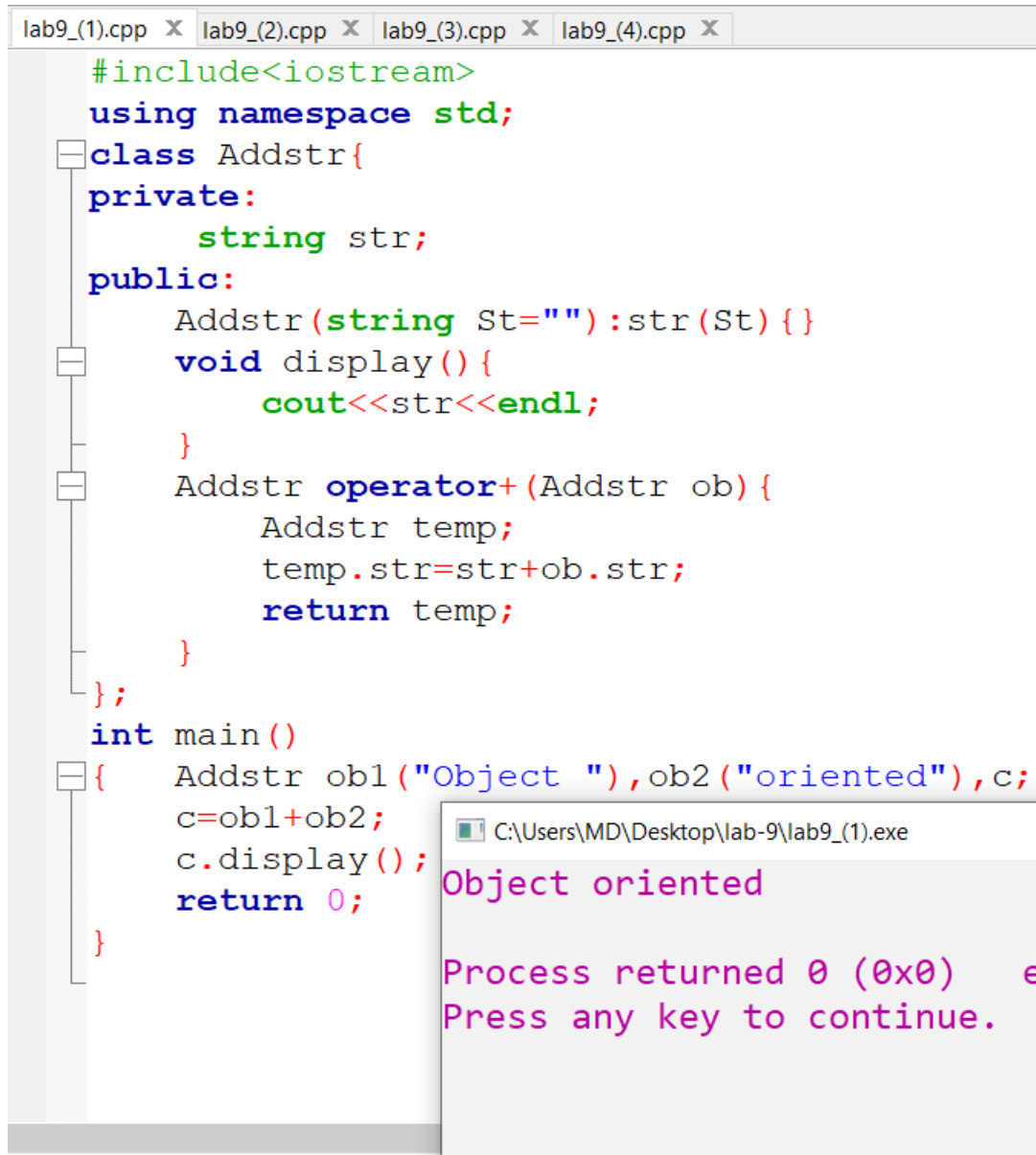


## LAB NO. 09

1. Overload the + for concatenating the two strings. For e.g "Object" + "oriented" = Objectoriented



```
lab9_(1).cpp X lab9_(2).cpp X lab9_(3).cpp X lab9_(4).cpp X
#include<iostream>
using namespace std;
class Addstr{
private:
    string str;
public:
    Addstr(string St=""):str(St){}
    void display(){
        cout<<str<<endl;
    }
    Addstr operator+(Addstr ob){
        Addstr temp;
        temp.str=str+ob.str;
        return temp;
    }
};

int main()
{
    Addstr ob1("Object "),ob2("oriented"),c;
    c=ob1+ob2;
    c.display();
    return 0;
}
```

C:\Users\MD\Desktop\lab-9\lab9\_(1).exe

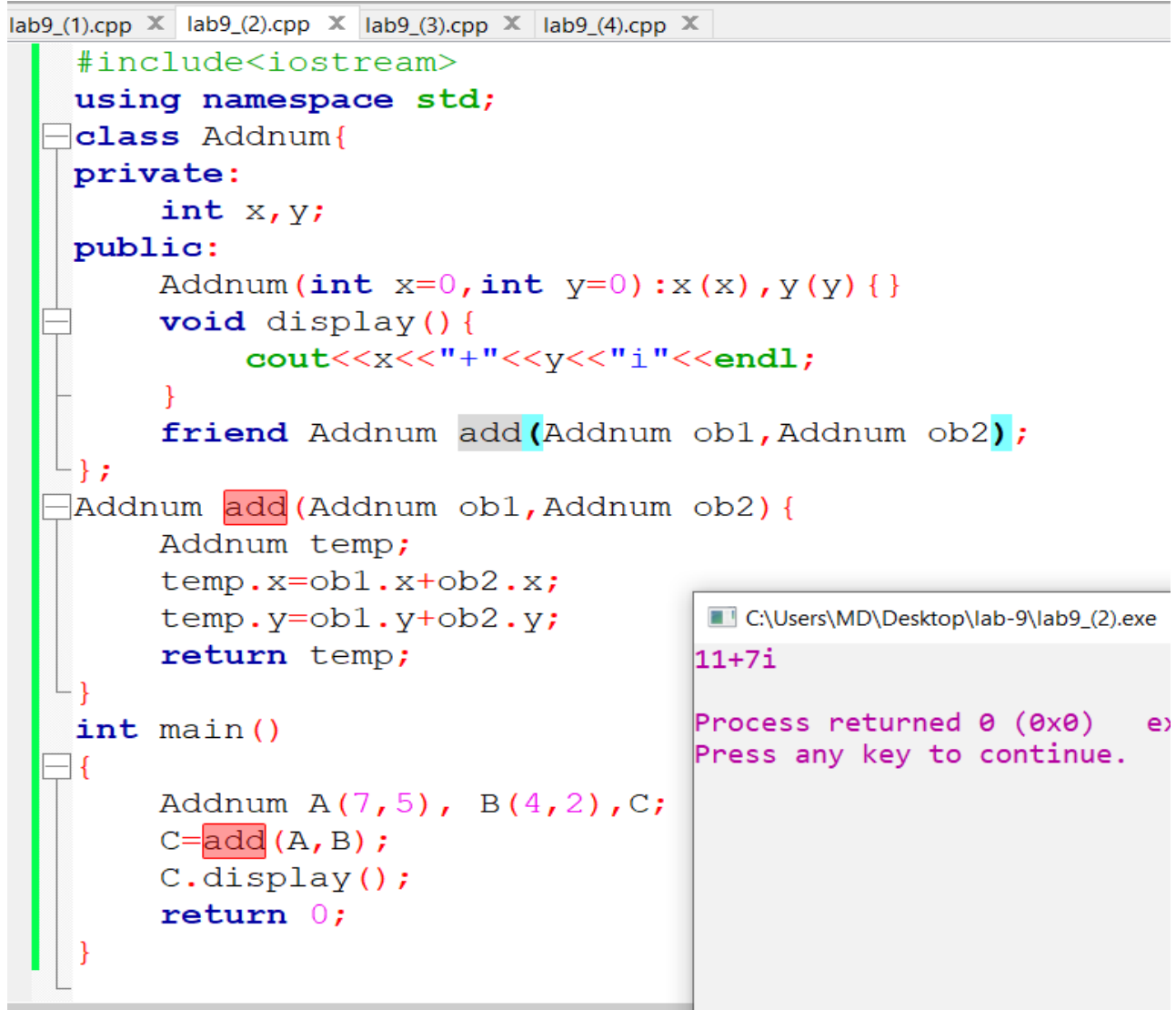
Object oriented

Process returned 0 (0x0) e

Press any key to continue.

## LAB NO. 09

2. Write a friend function for adding the two complex numbers, using single class.



```
#include<iostream>
using namespace std;
class Addnum{
private:
    int x,y;
public:
    Addnum(int x=0,int y=0):x(x),y(y){}
    void display(){
        cout<<x<<"+"<<y<<"i"<<endl;
    }
    friend Addnum add(Addnum ob1,Addnum ob2);
};
Addnum add(Addnum ob1,Addnum ob2){
    Addnum temp;
    temp.x=ob1.x+ob2.x;
    temp.y=ob1.y+ob2.y;
    return temp;
}
int main()
{
    Addnum A(7,5), B(4,2),C;
    C=add(A,B);
    C.display();
    return 0;
}
```

C:\Users\MD\Desktop\lab-9\lab9\_(2).exe

11+7i

Process returned 0 (0x0) e

Press any key to continue.

3. Overload the operator + for adding the timings (hr, min, sec) of two clocks.

## LAB NO. 09

```
lab9_(1).cpp x lab9_(2).cpp x lab9_(3).cpp x lab9_(4).cpp x
#include<iostream>
using namespace std;
class Time{
private:
    int hours, minutes, second;
public:
    Time():hours(0), minutes(0), second(0){}
    Time(int h, int mi, int se):hours(h), minutes(mi), second(se){}
    void display()
    {
        cout<<"hh:mm:ss "<<hours<<' '<<minutes<<' '<<second<<endl;
    }
    Time operator+(Time ob){
        int h=hours+ob.hours;
        int m=minutes+ob.minutes;
        int s=second+ob.second;
        m+=s/60;
        s%=60;
        h+=m/60;
        m%=60;
        return Time(h, m, s);
    }
}
```

```
lab9_(1).cpp x lab9_(2).cpp x lab9_(3).cpp x lab9_(4).cpp x
{
    cout<<"hh:mm:ss "<<hours<<' '<<minutes<<' '<<second<<endl;
}
Time operator+(Time ob){
    int h=hours+ob.hours;
    int m=minutes+ob.minutes;
    int s=second+ob.second;
    m+=s/60;
    s%=60;
    h+=m/60;
    m%=60;
    return Time(h, m, s);
}
};
int main()
{
    Time t1(8, 34, 55), t2(5, 42, 19), t;
    t=t1+t2;
    t.display();
    return 0;
}
```

C:\Users\MD\Desktop\lab-9\lab9\_(3).exe

hh:mm:ss 14 17 14

Process returned 0 (0x0) execution time  
Press any key to continue.

## LAB NO. 09

4. Create a class `BankAccount` representing a bank account with private members `balance`, `accountNumber`, and `customerName`. Implement a friend function `transferFunds()` that takes two `BankAccount` objects as arguments and transfers funds from one account to another. Also, make another class `BankManager` a friend of `BankAccount` class, which can access the `accountNumber` and `customerName` of any `BankAccount` object. Test your program by creating multiple bank accounts, transferring funds between them, and displaying account information using the `BankManager` class

```
(1).cpp x lab9_(2).cpp x lab9_(3).cpp x *lab9_(4).cpp x
#include<iostream>
using namespace std;
class BankAccount{
private:
    int balance,accountNumber;
    string customerName;
public:
    BankAccount():balance(0),accountNumber(0),customerName(""){}
    BankAccount(int b,int n,string name):balance(b),
                                                accountNumber(n),customerName(name){}
    friend void transferFunds(BankAccount &ob1,BankAccount &ob2);
    friend class BankManager;
};

void transferFunds(BankAccount &ob1,BankAccount &ob2){
    ob1.balance+=ob2.balance;
    ob2.balance-=ob2.balance;
}

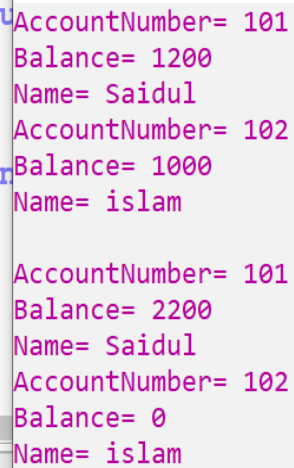
class BankManager{
public:
    void display(BankAccount &ob){
        cout<<"AccountNumber= "<<ob.accountNumber<<endl;
    }
}
```

## LAB NO. 09

```
class BankManager{
public:
    void display(BankAccount &ob){
        cout<<"AccountNumber= "<<ob.accountNumber<<endl;
        cout<<"Balance= "<<ob.balance<<endl;
        cout<<"Name= "<<ob.customerName<<endl;
    }
};

int main()
{
    BankAccount Ac1(1200,101,"Saidul"),Ac2(1000,102,"islam"),ac;
    BankManager ob;
    ob.display(Ac1);/// Before fu
    ob.display(Ac2);
    cout<<endl;
    transferFunds(Ac1,Ac2);
    ob.display(Ac1);/// After Fur
    ob.display(Ac2);

    return 0;
}
```



```
AccountNumber= 101
Balance= 1200
Name= Saidul
AccountNumber= 102
Balance= 1000
Name= islam

AccountNumber= 101
Balance= 2200
Name= Saidul
AccountNumber= 102
Balance= 0
Name= islam
```