

শিক্ষা নিয়ে গড়বো দেশ

তথ্য-প্রযুক্তির বাংলাদেশ

Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh



LAB REPORT-01

COURSE NO.- PROG 112

**COURSE TITLE- OBJECT ORIENTED
PROGRAMMING SESSIONAL**

SUBMITTED BY

Mehrin Farzana

ID: 2101013

Department of IRE

Session :2021-2022

Bangabandhu Sheikh Mujibur Rahman Digital
University, Bangladesh

SUBMITTED TO

Md.Toukir Ahmed

Lecturer

Department of IRE

Bangabandhu Sheikh Mujibur Rahman Digital
University, Bangladesh

Date of Submission: 16 July, 2023

Problem No.: 01

Problem Name: Classes and Objects.

Code:

```
#include<iostream>
using namespace std;

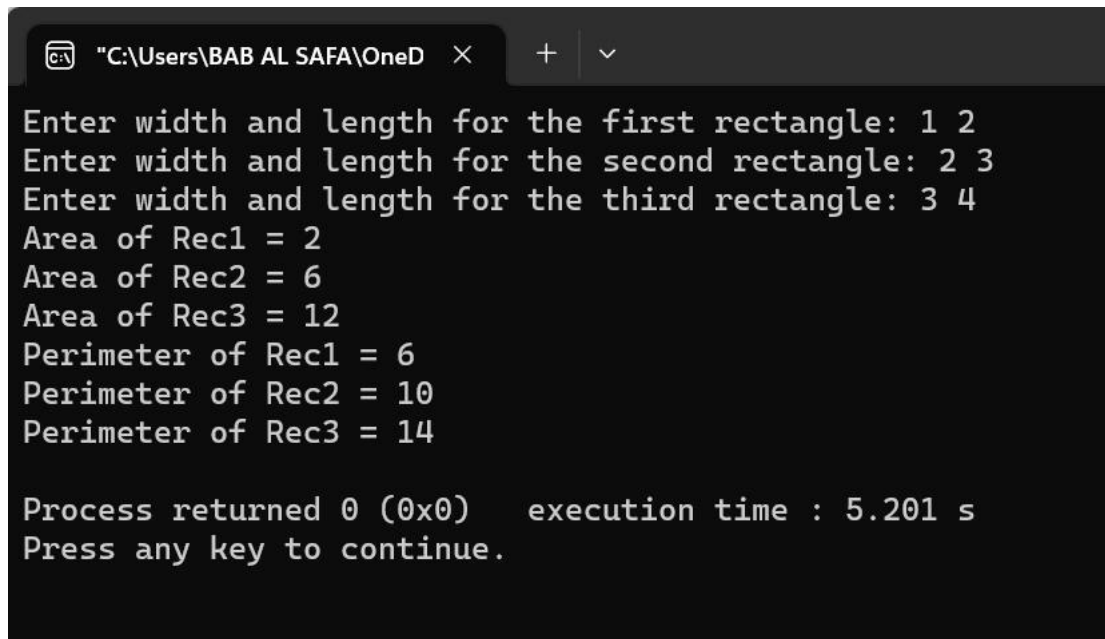
class Rectangle{
private:
    double w;
    double l;
public:
    Rectangle(){
        w=1.0;
        l=1.0;
    };
public:
    Rectangle(double width, double length){
        w = width;
        l = length;
    }
public:
    double area(){
        return w*l;
    }
public:
    double perimeter(){
        return 2*(w+l);
    }
};

int main(){
    double w1,l1, w2,l2, w3,l3;

    cout<<"Enter width and length for the first rectangle: ";
    cin>>w1>>l1;
    cout<<"Enter width and length for the second rectangle: ";
    cin>>w2>>l2;
    cout<<"Enter width and length for the third rectangle: ";
    cin>>w3>>l3;
    Rectangle r[3] = {Rectangle(w1,l1), Rectangle(w2,l2), Rectangle(w3,l3)};
    ;
    cout<<"Area of Rec1 = "<<r[0].area()<<endl;
    cout<<"Area of Rec2 = "<<r[1].area()<<endl;
```

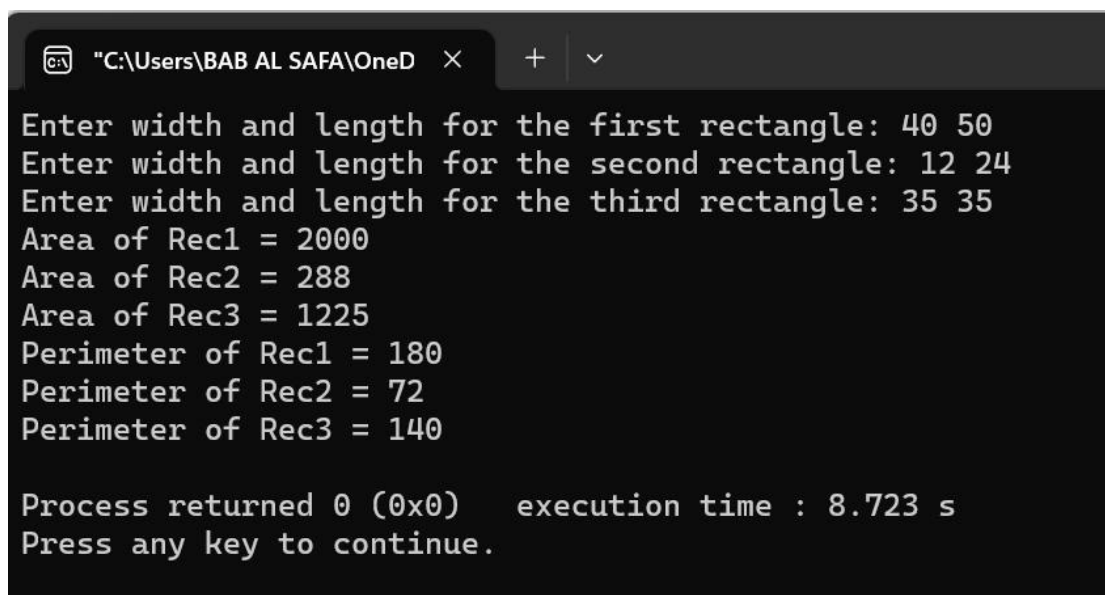
```
cout<<"Area of Rec3 = "<<r[2].area()<<endl;
cout<<"Perimeter of Rec1 = "<<r[0].perimeter()<<endl;
cout<<"Perimeter of Rec2 = "<<r[1].perimeter()<<endl;
cout<<"Perimeter of Rec3 = "<<r[2].perimeter()<<endl;
return 0;
}
```

Output:



```
"C:\Users\BAB AL SAFA\OneD" × + ∨  
Enter width and length for the first rectangle: 1 2  
Enter width and length for the second rectangle: 2 3  
Enter width and length for the third rectangle: 3 4  
Area of Rec1 = 2  
Area of Rec2 = 6  
Area of Rec3 = 12  
Perimeter of Rec1 = 6  
Perimeter of Rec2 = 10  
Perimeter of Rec3 = 14  
  
Process returned 0 (0x0) execution time : 5.201 s  
Press any key to continue.
```

Fig 1.1: Output on console for case 1.



```
"C:\Users\BAB AL SAFA\OneD" × + ∨  
Enter width and length for the first rectangle: 40 50  
Enter width and length for the second rectangle: 12 24  
Enter width and length for the third rectangle: 35 35  
Area of Rec1 = 2000  
Area of Rec2 = 288  
Area of Rec3 = 1225  
Perimeter of Rec1 = 180  
Perimeter of Rec2 = 72  
Perimeter of Rec3 = 140  
  
Process returned 0 (0x0) execution time : 8.723 s  
Press any key to continue.
```

Fig 1.2: Output on console for case 2.

Explanation:

This is a C++ program that calculates the area and perimeter of three rectangles. The program uses a class named Rectangle that has two private variables w and l for width and length respectively. The class has four public methods: a default constructor that sets w and l to 1.0, a constructor that takes two arguments for w and l, an area method that returns the area of the rectangle, and a perimeter method that returns the perimeter of the rectangle. The program then creates three Rectangle objects using the two-argument constructor and stores them in an array. Finally, it prints out the area and perimeter of each rectangle.

Problem No.: 02

Problem Name: Class and Objects.

Code:

```
#include<iostream>
#include <ctime>
#include <chrono>
using namespace std;

class Date {
private:
    std::chrono::time_point<std::chrono::system_clock>          start          =
std::chrono::system_clock::now();
    std::chrono::time_point<std::chrono::system_clock>          end            =
std::chrono::system_clock::now();
    std::time_t end_time = std::chrono::system_clock::to_time_t(end);
public:
    Date(){
        cout << "finished computation at " << std::ctime(&end_time)<<endl;
    }
};
```

```
class Account{
private:
    int ID;
    double Balance;
    double AnnualInterestRate;
    Date date_created;

public:
    Account(int id, double balance, double annualInterestRate){
        ID = id;
        Balance = balance;
        AnnualInterestRate = annualInterestRate;
    }

    bool withdraw(double amount){
        if(Balance-amount<0)
            return false;
```

```

        else{
            Balance-=amount;
            return true;
        }
    }

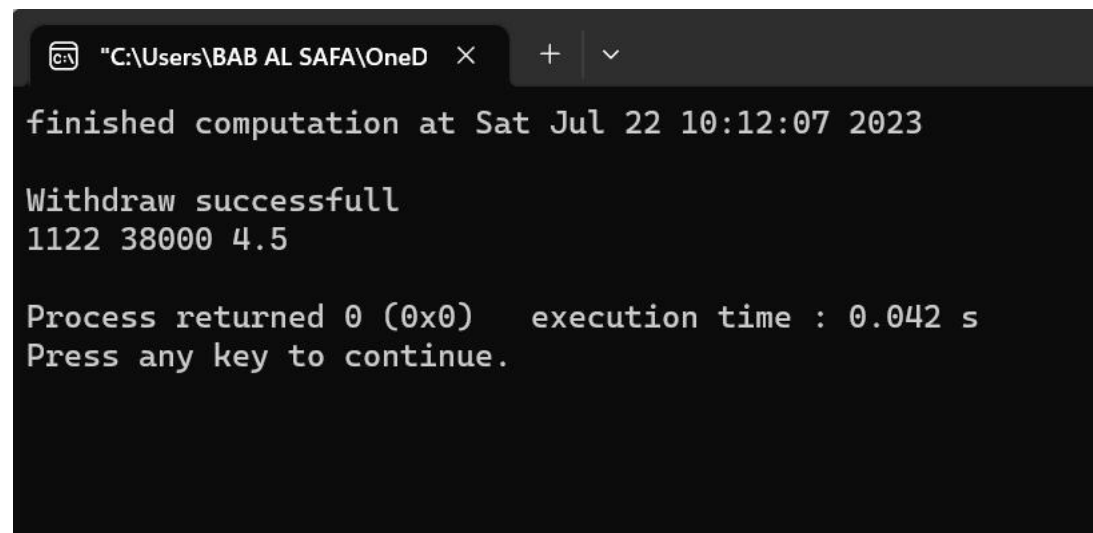
    void deposit(double amount){
        Balance+=amount;
    }
    void getId(){
        cout<<ID<<" ";
    }
    void getBalance(){
        cout<<Balance<<" ";
    }
    void getAnnualInterestRate(){
        cout<<AnnualInterestRate<<" ";
    }
};

int main(){
    Account testAccount(1122, 40000, 4.5);

    if(testAccount.withdraw(2500)){
        cout<<"Withdraw successfull"<<endl;
    }
    if(!testAccount.withdraw(2500)){
        cout<<"Withdraw not successfull"<<endl;
    }
    testAccount.deposit(3000);
    testAccount.getId();
    testAccount.getBalance();
    testAccount.getAnnualInterestRate();
    cout<<endl;
    return 0;
}

```

Output:



```
finished computation at Sat Jul 22 10:12:07 2023

Withdraw successfull
1122 38000 4.5

Process returned 0 (0x0)   execution time : 0.042 s
Press any key to continue.
```

Fig 2.1: Output on console.

Explanation:

This code creates a class called 'Account'. The class has four private data members: 'ID', 'Balance', 'AnnualInterestRate', and 'date_created'. The constructor initializes the data members with the values passed as arguments. The class has three public member functions: 'withdraw()', 'deposit()', and three accessor functions: 'getId()', 'getBalance()', and 'getAnnualInterestRate()'. The function 'withdraw()' checks if the amount to be withdrawn is greater than the balance. If it is, it returns false; otherwise, it subtracts the amount from the balance and returns true. The function 'deposit()' adds the amount passed as an argument to the balance. The accessor functions return the values of their respective data members. The code also creates an instance of the class called 'testAccount' with ID 1122, balance 40000, and annual interest rate 4.5. It then calls the member functions of this instance to test them. The code also creates an instance of the class called 'Date' which prints out the time when the computation finished.

Problem No.: 03

Problem Name: Class and Objects.

Code:

```
#include<iostream>
#include <string>
#include <list>
using namespace std;
class Account{
    private:
        int ID;
        double Balance;
        double AnnualInterestRate;

    public:
        Account(int id, double balance, double annualInterestRate){
            ID = id;
            Balance = balance;
            AnnualInterestRate = annualInterestRate;
        }

        bool withdraw(double amount){
            if(Balance-amount<0)
                return false;

            else{
                Balance-=amount;
                return true;
            }
        }

        void deposit(double amount){
            Balance+=amount;
        }
        int getID(){
            return ID;
        }
        string toString(){
            return  to_string(ID)  + " "  +  to_string(Balance)  + " "  +
to_string(AnnualInterestRate);
        }
};

class Client{
```

```

private:
    int id;
    std::string name;
    std::string phone;
    std::list<Account>accounts;

public:
    Client(int ID, std::string Name, std::string Phone){
        id = ID;
        name = Name;
        phone = Phone;
        std::list<Account>accounts;
    }

    bool addAccount(Account account){
        accounts.push_back(account);
        return true;
    }

    bool removeAccount(int accountId){
        int flag=0;
        for(auto it = accounts.begin(); it != accounts.end(); it++){
            if(it->getID() == accountId){
                flag=1;
                accounts.erase(it);
                break;
            }
        }
        if(flag)
            return true;
        else
            return false;
    }

    std::string toString(){
        string s = id + " " + name + " " + phone + "\n" ;
        for(auto it = accounts.begin(); it != accounts.end(); it++){
            s += it->toString() + "\n";
        }
        return s;
    }

    int getId(){
        return id;
    }

    void setId(int id){
        id=id;
    }

    string getName(){
        return name;
    }
}

```

```

    string setName(string name){
        name=name;
    }
    string getPhone(){
        return phone;
    }
    string setPhone(string phone){
        phone=phone;
    }
};

int main(){
    Client c[2] = {Client(100, "Ali", "123123123"), Client(200, "Deli", "456654546")};

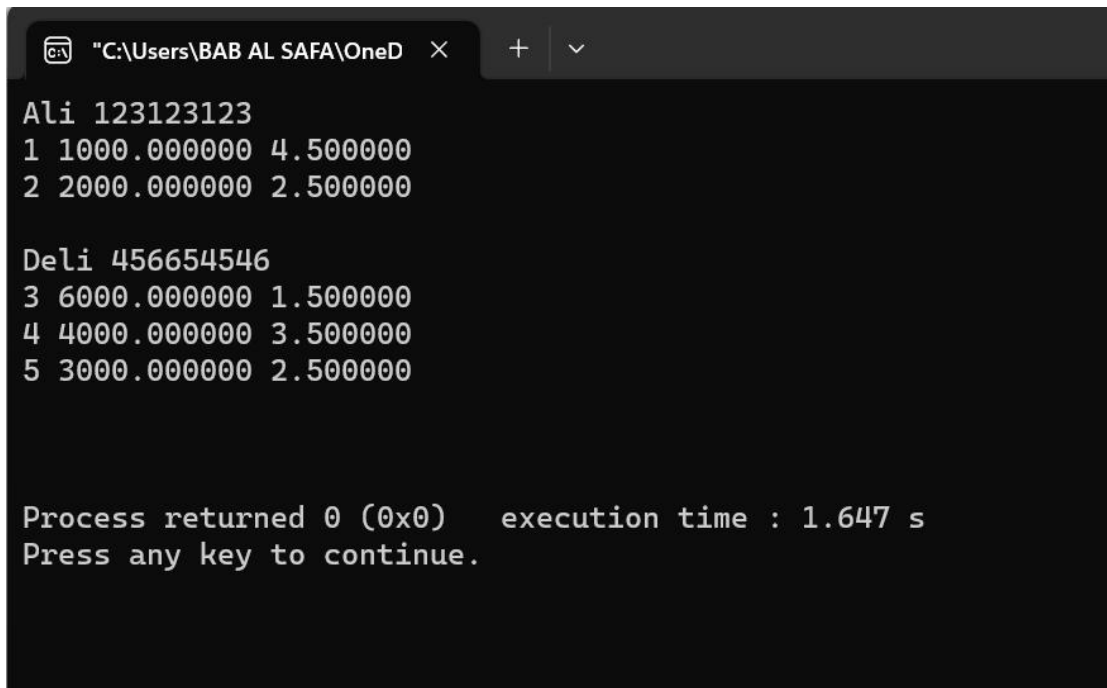
    c[0].addAccount(Account(1, 1000, 4.5));
    c[0].addAccount(Account(2, 2000, 2.5));

    c[1].addAccount(Account(3, 6000, 1.5));
    c[1].addAccount(Account(4, 4000, 3.5));
    c[1].addAccount(Account(5, 3000, 2.5));

    cout<<c[0].toString()<<endl;
    cout<<c[1].toString()<<endl;
    cout<<endl;
    return 0;
}

```

Output:

A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\BAB AL SAFA\OneD" with a close button (X) and window control buttons (+ and v). The command prompt has a black background with white text. The output is as follows:

```
Ali 123123123
1 1000.000000 4.500000
2 2000.000000 2.500000

Deli 456654546
3 6000.000000 1.500000
4 4000.000000 3.500000
5 3000.000000 2.500000

Process returned 0 (0x0)    execution time : 1.647 s
Press any key to continue.
```

Fig 3.1: Output on console.

Explanation:

The above code Creates two class called Account and Client. The class Client creates a list of type Account and performs operations such as adding and removing accounts from the list.

Then inside the main function it creates an array of type Client and adds two accounts to for the first client and three accounts for the second client and prints the string representation of the accounts onto the console.

Problem No.: 04

Problem Name: Function and function prototype

Code:

```
#include<iostream>
using namespace std;

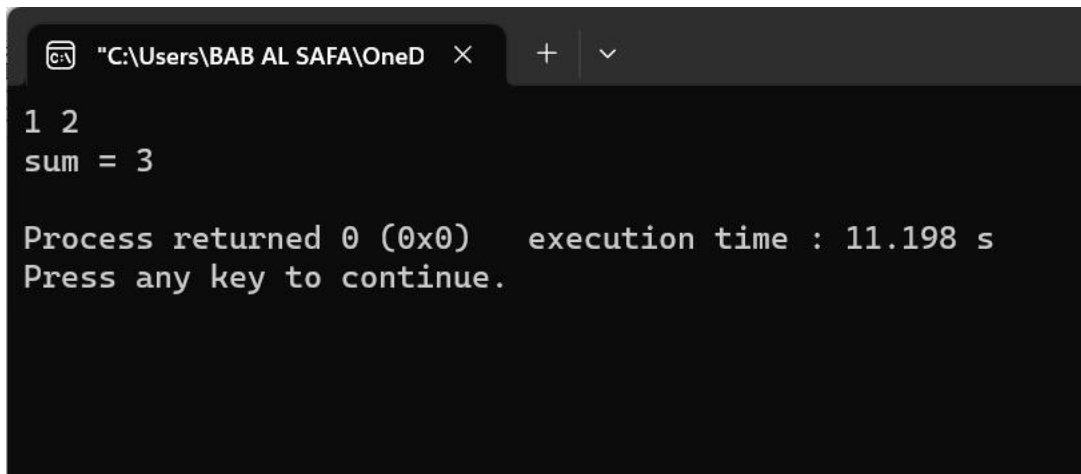
int sum(int, int);

int main(){
    int a,b;
    cin>>a>>b;
    cout<<"sum = "<<sum(a,b)<<endl;

    return 0;
}

int sum(int a, int b){
    return a+b;
}
```

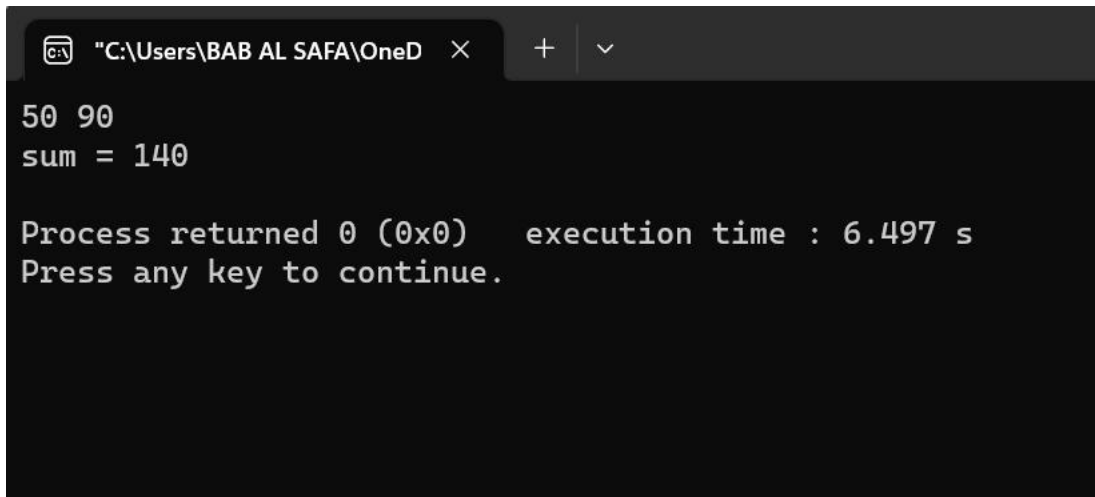
Output:



```
C:\Users\BAB AL SAFA\OneD × + v
1 2
sum = 3

Process returned 0 (0x0) execution time : 11.198 s
Press any key to continue.
```

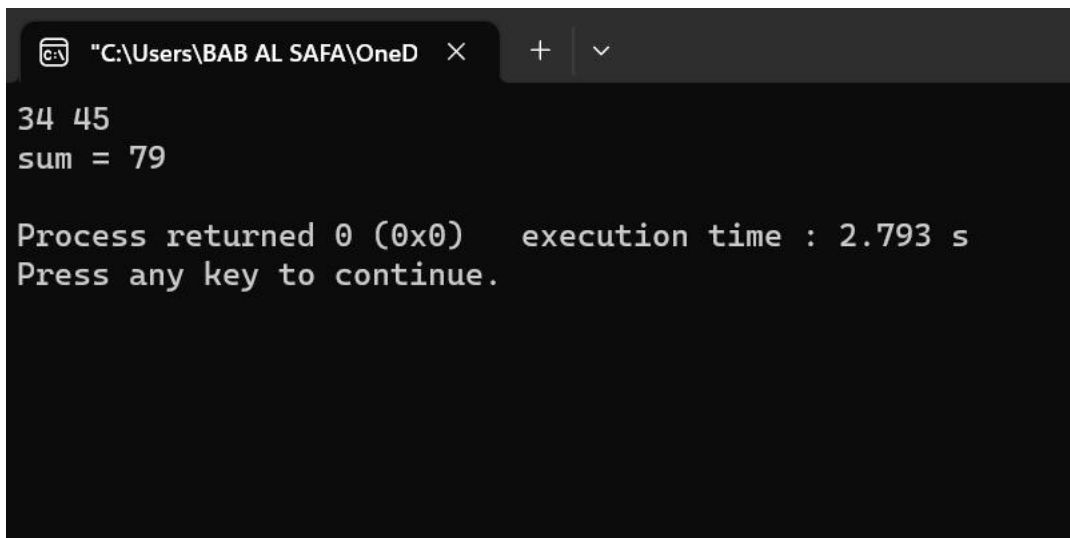
Fig 4.1: Output on console.



```
C:\Users\BAB AL SAFA\OneD × + v
50 90
sum = 140

Process returned 0 (0x0) execution time : 6.497 s
Press any key to continue.
```

Fig 4.2: Output on console.



```
C:\Users\BAB AL SAFA\OneD × + v
34 45
sum = 79

Process returned 0 (0x0) execution time : 2.793 s
Press any key to continue.
```

Fig 4.3: Output on console.

Explanation:

This code is a simple program that takes two integers as input and returns their sum.

The function sum takes two integers as input and returns their sum.

Before the main function the function prototype for function sum is declared.

The main function is the entry point of the program. It takes two integers as input using the cin function and stores them in variables a and b. It then calls the sum function with these two variables as arguments and prints the result using the cout function.

Problem No.: 05

Problem Name: Call by value and call by reference

Code:

```
#include<iostream>
using namespace std;

void swapByValue(int a, int b){
    int tmp = a;
    a = b;
    b = tmp;
}

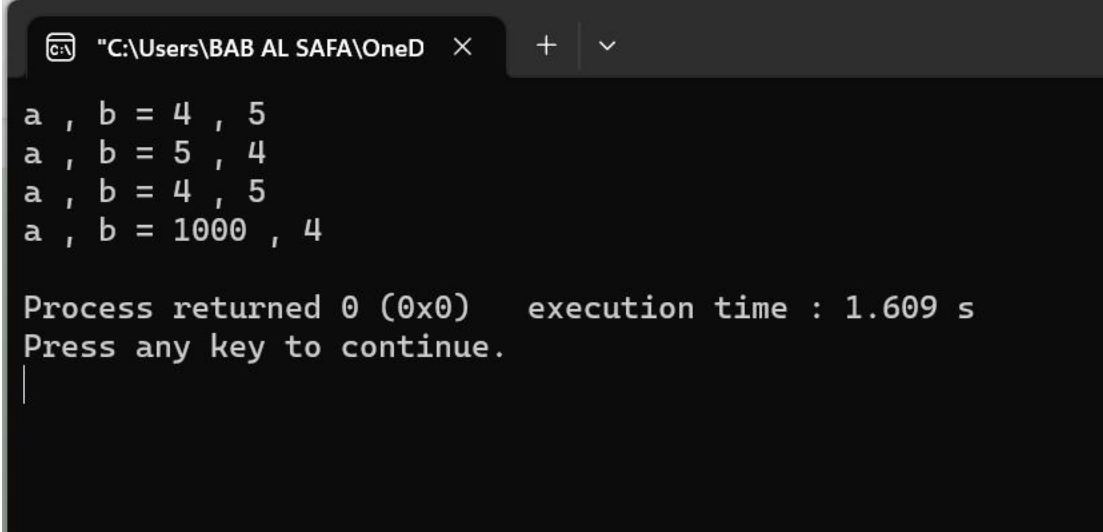
void swapByRef(int &a, int &b){
    int tmp = a;
    a = b;
    b = tmp;
}

void swapByRef2(int *a, int *b){
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

int &swapByRefVar(int &a, int &b){
    int tmp = a;
    a = b;
    b = tmp;
    return a;
}

int main(){
    int a=4,b=5;
    swapByValue(a,b);
    cout<<"a , b = "<<a<<" , "<<b<<endl;
    swapByRef(a,b);
    cout<<"a , b = "<<a<<" , "<<b<<endl;
    swapByRef2(&a,&b);
    cout<<"a , b = "<<a<<" , "<<b<<endl;
    swapByRefVar(a, b) = 1000;
    cout<<"a , b = "<<a<<" , "<<b<<endl;
    return 0;
}
```


Output:

A screenshot of a Windows console window. The title bar shows the file path "C:\Users\BAB AL SAFA\OneD" and standard window controls. The console output shows four lines of variable assignments: "a , b = 4 , 5", "a , b = 5 , 4", "a , b = 4 , 5", and "a , b = 1000 , 4". Below these, it says "Process returned 0 (0x0) execution time : 1.609 s" and "Press any key to continue." with a cursor on the next line.

```
"C:\Users\BAB AL SAFA\OneD" × + -  
a , b = 4 , 5  
a , b = 5 , 4  
a , b = 4 , 5  
a , b = 1000 , 4  
  
Process returned 0 (0x0) execution time : 1.609 s  
Press any key to continue.  
|
```

Fig 5.1: Output on console

Explanation:

This is a C++ code that demonstrates the difference between swapping by value and swapping by reference.

The code defines four functions that swap two integers. The first function `swapByValue` swaps two integers by passing them as values. The second function `swapByRef` swaps two integers by passing them as references. The third function `swapByRef2` swaps two integers by passing their pointers as references. The fourth function `swapByRefVar` swaps two integers by passing them as references and returning a reference.

The main function takes two integers as input from the user and calls all four functions to swap them.

When we pass variables by value, the function creates a copy of the variable and works on that copy. So when we swap variables using `swapByValue`, it only swaps the copies of the variables and not the original variables.

When we pass variables by reference, we pass the address of the variable to the function. So when we swap variables using `swapByRef`, it swaps the original variables.

When we pass pointers as references, we pass the address of the pointer to the function. So when we swap variables using `swapByRef2`, it swaps the values pointed to by the pointers.

When we pass variables as references and return a reference, we can use that reference to modify the original variable directly. So when we swap variables using `swapByRefVar`, it swaps the original variables.

Therefore:

- The first function `swapByValue` takes two integers as input and swaps them by creating copies of them.
- The second function `swapByRef` takes two integers as input and swaps them by passing their addresses as references.
- The third function `swapByRef2` takes two integer pointers as input and swaps them by passing their addresses as references.
- The fourth function `swapByRefVar` takes two integers as input and swaps them by passing their addresses as references and returning a reference.

Problem No.: 06

Problem Name: Static Data members and Methods

Code:

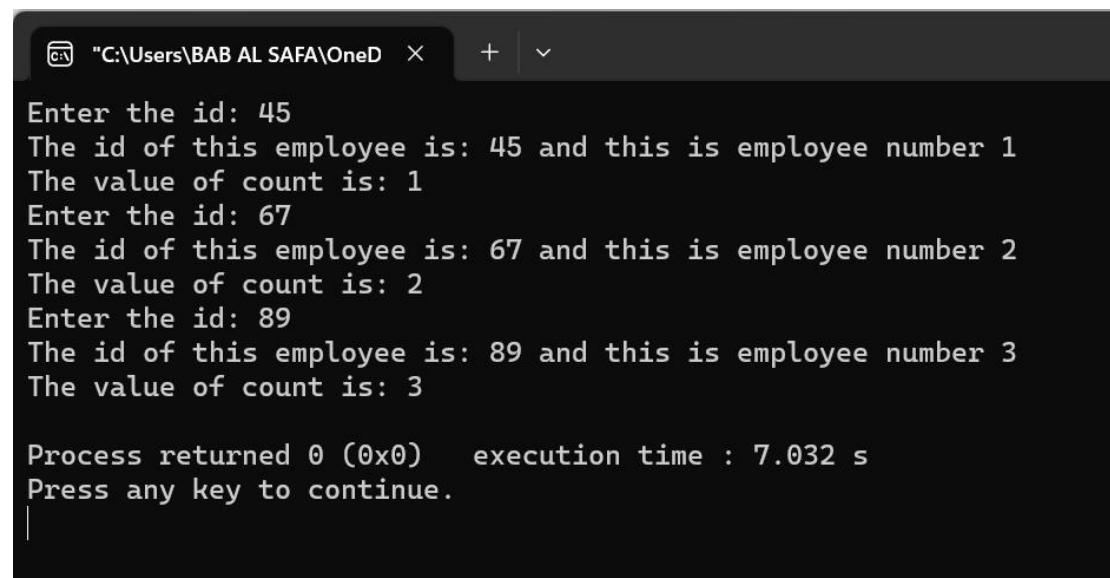
```
#include<iostream>
using namespace std;

class Employee{
private:
    int id;
    static int count_;
public:
    void setData(){
        cout<<"Enter the id: ";
        cin>>id;
        count_++;
    }
    void getData(){
        cout<<"The id of this employee is: "<<id<<" and this is employee number
"<<count_<<endl;
    }
    static void getCount(){
        cout<<"The value of count is: "<< count_<<endl;
    }
};

int Employee::count_;
int main(){
    Employee harry, rohan, lovish;

    harry.setData();
    harry.getData();
    Employee::getCount();
    rohan.setData();
    rohan.getData();
    Employee::getCount();
    lovish.setData();
    lovish.getData();
    Employee::getCount();
    return 0;
}
```

Output:



```
"C:\Users\BAB AL SAFA\OneD"
Enter the id: 45
The id of this employee is: 45 and this is employee number 1
The value of count is: 1
Enter the id: 67
The id of this employee is: 67 and this is employee number 2
The value of count is: 2
Enter the id: 89
The id of this employee is: 89 and this is employee number 3
The value of count is: 3

Process returned 0 (0x0)   execution time : 7.032 s
Press any key to continue.
```

Fig 6.1: Output on console

Explanation:

This is a C++ code that demonstrates the use of classes and static variables.

The code defines a class Employee that has two private data members - id and count_. The class has three member functions - setData, getData, and getCount.

The setData function takes input from the user for the id data member and increments the static variable count_.

The getData function prints the value of the id data member and the value of the static variable count_.

The getCount function prints the value of the static variable count_.

The main function creates three objects of the class Employee - harry, rohan, and lovish. It then calls the setData, getData, and getCount functions for each object.

Therefore:

- The class Employee has two private data members - id and count_.
- The class has three member functions - setData, getData, and getCount.
- The setData function takes input from the user for the id data member and increments the static variable count_.
- The getData function prints the value of the id data member and the value of the static variable count_.
- The getCount function prints the value of the static variable count_.
- The main function creates three objects of the class Employee - harry, rohan, and lovish.
- It then calls the setData, getData, and getCount functions for each object