Experiment Number: 01 and 02

Aim: Implementation of Object Oriented Concepts of data encapsulation using C++ programs.

Lab Objective: Understand data abstraction and object orientation

Lab Outcome Mapped: Implement Object Oriented concepts in C++. (LO1)

Requirements: Any text editor to be able to edit C++ code and any compiler for C++ to run and execute the code.

Theory:

C++ extends C with classes for implementing user defined data types. The important features of C++ programming language are –

- C++ is an Object Oriented Programming (OOP) language
- It is platform/machine independent
- It is an imperative programming language
- C++ provides a lot of in-build functions that makes the development faster.

One of the important concept that makes C++ OOP is concept of class. It is a way to bind the data and its associated member functions together. It allows the data and functions to be hidden, if necessary from external use. When defining a class, new abstract data type gets created.

The class specification has two parts-

- Class declaration
- Class function definitions

The general form of class declaration is-

```
Class class_name {

private:
data_type member_var_name;
…
public:
return_type mem_function_name(parameter_list);
...
};
```

The keyword class specifies, that what follows is an abstract data of type class_name. The body of a class is enclosed within braces and terminated by a semicolon. The class body contains the declarations of variables and functions. These functions and variables are collectively called class members. The keywords private and public are known as visibility labels. The default visibility access is private. The class members that have been declared as private can be accessed only from within the class. On the other hand, public members can be accessed from outside the class also. The data hiding (using private declaration) is the key feature of object oriented programming If both the variables and members are declared as private then such a class is completely hidden from the outside world and does not serve any purpose. The variables declared inside the class are known as data members and functions are known as member functions. The member functions can be declared both inside and outside the class. The binding of data and function together into a single class type variable is referred to as encapsulation. The member data and functions can be accessed using objects. Syntax for creating objects is similar to that of creation of a variable of perticular type. We can sonsider Class to be a derivered data type defined by user which not only encapsulates data but also has set of member functions which can handle data manipulations. Objects are instances of class that can be used by programmer as perhis requirement.

Performance steps:

1. Students are made aware of the programming environment they will be working in.
2. They are taught how to access, read and edit a c++ code, compile it using g++ to produce executable output. Finally they are taught to execute the code to know the output.
3. Then students are provided with sample codes presented in **Program Code** and asked to understand, execute the same and note the output.
4. Students are walked thourgh the tutorial on inheritance from online resource Inheritance Tutorial.
5. Students are asked to type in any one of the examples from tutorial on their machines and understand code and execute it.

Post Experimental Exercise:(To be completed by students in writing as submission)

1. What is a class? How does it accomplish data hiding?
2. What are objects? How are they created?
3. What is a function overloading? Which program code achieves it?
4. WAP to define a class to represent bank account. Include the following members-

Data members- Name of depositor, account number, type of account, balance amt. in account
Member functions- To assign initial values, to deposit an amount, to withdraw an amount after checking the balance, to display name and balance. Write sample main code to demonstrate creation of object and use of member variables and functions.

5. Name the type of inheritance supported by C++. Explain each in one or two lines.

Program Code:

Program Code 1 : C++ Program to add two number using function.

```
#include<iostream>

using namespace std;
int add (int a, int b)
{
int r;
r=a+b;// a and b will be fetched from main()
return r; //since the return type is int
}
int main()
{
int z,a,b;
cout<<"Enter the two numbers"<<endl;
cin>>a>>b;
z=add(a,b);
cout<<"The result is="<<z;
return 0;
}
```

Program Code 2 : C++ Program to learn how to pass reference to a function.

```
//program to study pass values by reference

#include <iostream>
using namespace std;
void fun1(int *ptr)
{
  *ptr=20;
}
int main()
{
  printf("I am the main");
  int x=30;
  cout<<"The value of x before call"<<" "<<x<<endl;
  cout<<"The memory address location of x before is"<<" "<<(&x)<<endl;
  fun1(&x);
  cout<<"The memory address location of x after is"<<" "<<(&x)<<endl;
  cout<<"The value of x after call is"<<" "<<x<<endl;
  return 0;
}
```

The function fun1() expects a pointer ptr to an integer (or an address of an integer). It modifies the value at the address ptr. The dereference operator * is used to access the value at an address. In the statement '*ptr = 30', value at address ptr is changed to 30. The address operator & is used to get the address of a variable of any data type. In the function call statement 'fun(&x)', the address of x is passed so that x can be modified using its address.

Program Code 2 : C++ Program to learn concept of function overloading.

```
#include<iostream>
using namespace std;
int plusFunc(int x, int y) { return x + y; }
double plusFunc(double x, double y) { return x + y; }
float plusFunc(float x, int y) { return x+y; }
int main() {
   int myNum1 = plusFunc(8, 5);
   double myNum2 = plusFunc(4.3, 6.26);
   float mynum3=plusFunc(4.6,7);
   cout << "Int: " << myNum1 << "\n"; cout << "Double: " << myNum2;
   return 0;
}
```

Program Code 3 : C++ Program to learn data encapsulation

```
#include<iostream>
using namespace std;
//Class definition
class item {
        private:
            int a; int b;

        public:
            void getdata(int,int);
            void putdata(void);
};
void item:: getdata(int x, int y) {
          a=x; //private members can be directly accessed by the member functions
          b=y;
}
void item:: putdata() {
           cout<<"The value of a is="<<a<<endl; cout<<"The value of b is="<<b<<endl;
}
int main() {
            item t; //Create object of class
            t.getdata(100,200);
            t.putdata();
            return 0;
}
```

Conclusion:

Thus we have learned about important feature of OOP using C++ as OOP language.

Reference:

1. E Balaguguswamy, “Object Oriented Programming with C++”, second edition Tata McGraw Hill (Chapter 5).
2. Online Tutorial on C++ inheritance, https://www.javatpoint.com/cpp-inheritance, access date: 30/11/2020.