

Himalaya College of Engineering

**Advanced C++ Programming Lab Report**

Lab 4: Concepts in OOP (Object as Function arguments,Arrays, Pointers, DMA, Static and Constant Member Function and Friend Function.

**Prepared By :** Monsoon Sapkota (HCE081BEI022)

**Subject :** Object-Oriented Programming (OOP)

**Program :** Bachelor of Electronics, Communication and Information Engineering

**Institution :** Himalaya College of Engineering

**Date :** June 06, 2025

# OBJECTIVE

* To understand the concept of classes and objects in C++.
* To use object as a datatype.
* To use DMA for objects.
* Understand Static and Constant member and member functions.
* To understand and use Friend function and friend classes.

# BACKGROUND THEORY

**A. Object as Function Arguments**

In C++ we are able to use object as a datatype (argument) by following ways:

* Pass by Value: - It copies the object, changes made in the copy doesn’t affect the original value.
* Pass by Reference: - In it the object is passed by a reference in which the change made is reflected back on the original value.
* Pass by Pointer: - In it address is passed by which it is useful for Dynamic memory and null checks.

**Example:**

void display(MyClass obj); // By value void display(MyClass &obj); // By reference void display(MyClass \*obj); // By pointer

1. **Array of Objects**

An array of objects refers to a group of objects of the same class stored in contiguous memory locations.

**Example:**

Student students[5]; //Array of students with 5 students.

1. **Pointer to objects**

In C++, a pointer to an object is a pointer that stores the memory address of an object. This allows you to dynamically allocate objects, pass objects to functions, or manipulate them indirectly.

**Example:**

MyClass obj; // Regular object

MyClass\* ptr = &obj; // Pointer to the object

1. **Dynamic Memory Allocation**

It helps us to allocate memory during runtime rather than during compiling. This is particularly useful when the amount of memory required isn’t known to us.

**Example:**

class Student { public:

void show() {

cout << "Student object created dynamically." << endl;

}

};

Student \*s = new Student(); // Dynamically created object s->show();

delete s; // Memory deallocated

**E. Dynamic Constructors**

A Dynamic Constructor in C++ is a constructor that allocates memory at runtime using dynamic memory allocation (DMA) usually through the new operator. This is useful when the size of the memory or data is not known during compile time and needs to be determined during program execution.

**Example:**

class DynamicArray {

int\* arr; int size; public:

DynamicArray(int s) {

size = s;

arr = new int[size]; // Dynamic memory allocation

}

~DynamicArray() {

delete[] arr; // Freeing memory

}

**F. Static Data Member**

A static data member in C++ is a class variable that is shared by all objects of the class. Unlike normal data members, only one copy of the static member exists, regardless of how many objects are created. It is declared using static keyword.

**Example:**

#include <iostream> using namespace std; class Test { static int count; public:

Test() { count++; }

void show() { cout << "Count: " << count << endl; }

};

int Test::count = 0;

int main() { Test a, b;

a.show(); // Output: Count: 2 return 0;

}

**G. Static Member function**

A static member function is a function that belongs to the class, not to any specific object. It can only access static data members of the class and can be called using the class name.

**Example:**

#include <iostream> using namespace std;

class Demo {

static int val; public:

static void display() {

cout << "Value: " << val << endl;

}

};

int Demo::val = 10;

void main() {

Demo::display(); // Output: Value: 10

}

**H. Constant Member Function**

A constant member function is a function that does not modify any member variables of the class. It is declared by adding the const keyword after the function signature.

**Example:**

#include <iostream> using namespace std;

class Sample {

int data; public:

Sample(int x) { data = x; } void show() const {

cout << "Data: " << data << endl;

}

};

void main() { Sample obj(50);

obj.show(); // Output: Data: 50

}

**I. Constant Objects**

A constant objectin C++ is declared using the **const** keyword. Once created, its data members cannot be modified, and it can only call **const** member functions.

**Example:**

#include <iostream>

using namespace std;

class Item {

public:

void display() const {

cout << "Constant object calling const function." << endl;

} };

void main() {

const Item obj; // Constant object

obj.display(); // Only const functions can be called

}

**J. Friend Functions**

A friend function in C++ is a function that is not a member of a class but is granted special access to its private and protected members. It is declared using the friend keyword inside the class.

**Example:**

#include <iostream>

using namespace std;

class Box {

private:

int length;

public:

Box() : length(10) {}

friend void display(Box); // Friend function declaration

};

void display(Box b) {

cout << "Length: " << b.length << endl; // Access private data

}

void main() { Box b;

display(b); // Output: Length: 10

}

**K. Friend Classes**

A friend class in C++ is a class that is granted access to the private and protected members of another class. One class can declare another class as its friend, allowing all the member functions of the friend class to access its private data.

**Example:**

#include <iostream> using namespace std; class ClassB; // Forward declaration class ClassA { int dataA; public:

ClassA(int x) : dataA(x) {} friend class ClassB; // ClassB is a friend of ClassA

}; class ClassB { public:

void showData(ClassA &a) { cout << "Data from ClassA: " << a.dataA << endl; // Access private member of ClassA } }; void main() {

ClassA objA(100); ClassB objB; objB.showData(objA); } // Output: Data from ClassA: 100