HIMALAYA COLLEGE OF ENGINEERING

Advanced C++ Programming Lab Report

Lab 2: Functions, Structures, and Memory in C++

Prepared By: Mukesh Pandeya

Subject: Object Oriented Programming (OOP)

Program: Bachelors of Electronics and Computer Engineering

Institution: Himalaya College of Engineering

Date: June 8, 2025

Objectives:

- To understand and implement function overloading in C++.
- To utilize inline functions for efficiency.
- To work with default arguments in functions.
- To implement call-by-reference.
- To manipulate arrays and pointers.
- To differentiate between structures and unions.
- To understand enumeration types in C++.
- To dynamically allocate and deallocate memory.

Tools and Libraries Used:

- Programming Language: C++
- IDE: G++
- Libraries: include <iostream>, include <string>

Theory:

Function Overloading

Function overloading allows multiple functions to have the same name with different parameters. The compiler determines which function to invoke based on the function signature.

Example:

```cpp
int add(int a, int b);
float add(float x, float y);
float add(int a, float b);
```

Inline Functions

Inline functions are used to reduce the overhead of function calls. When a function is marked as inline, the compiler attempts to expand it at the point of call.

Example:

```
inline int square(int n);
```

Default Arguments

Default arguments are specified in function declarations and allow functions to be called with fewer arguments than declared.

Example:

```
float calculateTotal(float price, int quantity = 1);
```

Call-by-Reference

Using reference variables in function parameters enables the function to modify the original values passed to it.

Example: i) Call by refrence

```
void swapNumbers(int &a, int &b);
```

i) Return refrence

```
int& getElement(int arr[], int index);
```

Pointers and Arrays

Pointers can access and manipulate array elements directly using pointer arithmetic.

Example:

```
int* ptr = arr;
```

Structures vs Unions

Structure: Allocates separate memory for each member.

Example:

```
struct StdStructure {
  int roll;
  string name;
  float marks;
};
```

Union: Allocates shared memory, and only one member can be used at a time.

Example:

```
union StdUnion {
  int roll;
  string name;
  float marks;
};
```

Enumerations

Enums allow defining a set of named integral constants to make code more readable and maintainable.

Example:

```
enum Day { Sunday, Monday, ... };
```

Dynamic Memory Allocation

Using new and delete operators in C++, memory can be allocated and deallocated at runtime.

Example:

```
int* arr = new int[n];
delete[] arr;
```