**Objectives:**

* 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:**

**Overloading of Functions**

Function overloading enables defining multiple functions with identical names but varying parameter lists. The compiler determines which function to invoke based on the function signature.

Illustration:

int add(int a, int b);

float add(float x, float y);

float add(int a, float b);

**Inline Functions**

Inline methods 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.

Illustration:

inline int square(int n);

**Default Arguments**

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

Illustration:

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

**Pass-by-reference**

Reference variables in function arguments allow modifying the passed values directly passed to it.

Illustration:

**i) Call by reference**

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

**ii) Return reference**

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

**Arrays and pointer usage**

Pointers enable direct manipulation and access of array data through arithmetic operations.

Illustration:

int\* ptr = arr;

**Difference between Structures and Unions**

**Structure:** Each field is assigned its own memory location.

Illustration:

struct StdStructure {

int roll;

string name;

float marks;

};

**Union:** Shares memory among members, allowing access to one field at a time.

Illustration:

union StdUnion {

int roll;

string name;

float marks;

};

**Enumeration types**

Enums define a group of named constants with integral values to make code more readable and maintainable.

Illustration:

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

**Memory allocation and deallocation at runtime**

C++ allows runtime memory management through the use of new and delete operators.

Illustration:

int\* arr = new int[n];

delete[] arr;