VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANASANGAMA, BELAGAVI - 590018

Inter/Intra Institutional Internship Report

On

Data Structures and Object-Oriented Programming Concepts

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering

in

Artificial Intelligence and Machine Learning

Submitted by

1.	1RN21AI020	ANAND OKADE
2.	1RN21AI051	HARSH MOTWANI
3.	1RN21AI054	HARSHITH S GOWDA
4.	1RN21AI061	KHUSHI ETAGI

RNS INSTITUTE OF TECHNOLOGY

(AICTE Approved, **VTU** Affiliated and **NAAC 'A+'** Accredited)
(**UG programs** – CSE, ECE, ISE, EIE and EEE are **Accredited** by **NBA** up to **30.6.2025)**
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098

Department of AI & ML

2022 – 2023

RNS INSTITUTE OF TECHNOLOGY

(**AICTE** Approved, **VTU** Affiliated and **NAAC 'A+'** Accredited)
(**UG programs** – CSE, ECE, ISE, EIE and EEE are **Accredited** by **NBA** up to **30.6.2025)**
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098

Department of AI & ML

CERTIFICATE

Certified that **Anand Okade-1RN21AI020, Harsh Motwani-1RN21AI051, Harshith S Gowda-1RN21AI054 , Khushi Etagi -1RN21AI061** has carried out the Internship on **Data Structures** and **Object Oriented Programming Concepts** conducted internally from **11th October 2022 to 31st October 2022**. Candidate has fulfilled all the requirements prescribed in the curriculum. Candidate has incorporated all the corrections/suggestions indicated during the Internal Assessment. The internal internship work has been approved as it satisfies the curriculum requirements.

Internship Co-ordinator	**Guide**	**HoD**
Ms. Sajitha N	Dr. Rama Satish K V	Dr. Harsha S
Assistant Professor	Associate Professor	Department of AI & ML
Department of AI&ML	Department of AI&ML	RNSIT, Bengaluru
RNSIT, Bengaluru	RNSIT, Bengaluru	

Name & Signature

Examiner 1:

Examiner 2:

ABSTRACT

When was the last time you wished to invest in the stock market? Or wish to increase your money in a faster way than the slow and time consuming bank FDs? The market maybe currently underperforming but there is definitely some light to look for in the darkness and we will help you look for it with our project.

In our project, we are greeting the user and asking them if they want to provide their details in regards to the amount of money they are willing to invest. Then, we will show them the various available stocks that can be invested in. Then we will ask the user which stocks they have invested in and then run a predictive analysis of its performance over a period of time.

Swapping of two numbers is a common operation in programming. It involves exchanging the values of two variables. In C++, we can perform swapping of two numbers using dynamic memory allocation.

Dynamic memory allocation is a technique in C++ that allows us to allocate memory at runtime. It enables us to allocate memory to variables dynamically as per our requirement.

To swap two numbers using dynamic memory allocation in C++, we can follow the steps below:

Declare two integer pointers: "ptr1" and "ptr2".

Use the "new" operator to dynamically allocate memory for the two integers.

Prompt the user to enter the values of the two numbers.

Store the values entered by the user in the dynamically allocated memory using the pointers.

Use a temporary variable to swap the values stored in the dynamically allocated memory.

Display the swapped values of the two numbers.

ACKNOWLEDGEMENT

At the very onset, we would like to place on record our gratitude to all those people who have helped us in making this project work a reality. Our Institution has played a paramount role in guiding us in the right direction.

We would like to profoundly thank **Sri. Satish R Shetty**, Chairman, RNS Group of Institutions. Bangalore for providing such a healthy environment for the successful completion of this Internship.

We would also like to thank our beloved Principal, **Dr. M K Venkatesha**. for providing the necessary facilities to carry out this work.

We are extremely grateful to **Dr. Harsha S**, Head of the Department of Artificial Intelligence and Machine Learning, for having accepted to guide me in the righirection with all his wisdom.

We would like to express our sincere thanks to our Internship co-ordinator **Ms. Sajitha N** Assistant Professor, Department of Artificial Intelligence and Machine Learning for her constant encouragement that motivated us for the successful completion of this Internship.

We would like to thank **Dr. Rama Satish K V,** Associate Professor, Department of Artificial Intelligence and Machine Learning for his continuous guidance and constructive suggestions for this Internship.

Last but not the least. we are thankful to all the teaching and non-teaching staff members of the Artificial Intelligence and Machine Learning Department for their encouragement and support throughout this work.

Signature

Anand Okade 1RN21AI020

Harsh Motwani 1RN21AI051

Harshith S Gowda 1RN21AI054

Khushi Etagi 1RN21AI061

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

LEARNING OBJECTIVES

This C and C++ internship can be a valuable learning opportunity for us to gain practical experience in software development using these programming languages. The objectives of such an internship can vary depending on the specific program and the company providing the opportunity. The goals include gaining hands-on experience with coding and debugging C and C++ programs, learning to use development tools and technologies, improving coding skills and software development practices, and working collaboratively with other developers on projects.

In addition to technical skills, an internship in C and C++ programming can also help us develop valuable professional skills, such as communication, teamwork, and project management. Working collaboratively with other developers on projects, interns will learn how to communicate effectively with team members, share ideas and best practices, and work towards common goals. They will also gain experience in managing their time and workload, working with project managers and other stakeholders to deliver high-quality software on time and within budget.

Overall, a C and C++ internship can be an excellent way to gain practical experience in software development, build technical skills, and develop valuable professional skills that will help participants succeed in their future careers. By the end of the program, participants should have a solid foundation in C and C++ programming, as well as practical experience working on real-world projects in a professional environment.

Chapter 2

WEEKLY OVERVIEW OF INTERNSHIP

C Programming

Session	Topic
1	**Introduction** Algorithms data types conditional statements loops in c jumps in c
2	**Functions** user defined functions types of user defined functions recursion examples of recursion
3	**Arrays** 1D arrays 2D arrays multi-dimensional arrays searching and sorting using 1D arrays passing array as parameter to functions
4	**Strings** string operations string operations using user defined functions built in string functions
5	**Pointers** pointers and arrays memory allocation pointers and functions example programs using pointers
6	**Structure** Array of structure struct and pointers union struct and functions Example programs using structure

Introduction to Object Oriented Programming

Session	Topic
7	**Introduction to C++** datatypes basic input, and output operators conditional statements loops
8	**C++ functions** function overloading default arguments storage class recursion return reference C++ arrays and strings Structure and functions
9	**C++ classes and objects** Constructors Objects and functions operator overloading
10	**C++ inheritance** function overriding friend function

Introduction:

C is a general-purpose, high-level programming language that was first developed in the early 1970s by Dennis Ritchie at Bell Labs. It is widely used for system programming, operating systems, embedded systems, and game development. C is a compiled language, which means that code written in C needs to be compiled before it can be executed. It is a low-level language, which gives programmers direct control over the computer's hardware. C is known for its speed, efficiency, and portability. It has influenced many other programming languages such as C++, Java, and Python. C has a relatively simple syntax and a small set of keywords, which makes it easy to learn and use for beginners. C is still widely used today and is considered an essential language for programmers.

Algorithms:

Algorithms are step-by-step procedures used to solve problems or perform a specific task. They are a set of rules or instructions that guide a computer program to complete a particular task efficiently and effectively. In C programming, algorithms are used to solve a wide range of problems, from simple arithmetic calculations to complex data analysis and processing. Programmers use algorithms to design and implement programs that can solve real-world problems**.**

Data types:

Data types in C programming are used to define the type of data that a variable can hold. A variable is a container that stores data in the memory of the computer. In C programming, there are four main data types: integers, floating-point numbers, characters, and booleans. These data types are used to define the type of data that can be stored in a variable and the operations that can be performed on the data. For example, integers are used to store whole numbers, floating-point numbers are used to store real numbers, characters are used to store letters and symbols, and booleans are used to store true or false values.

Conditional Statements:

Conditional statements in C programming are used to make decisions based on the value of a variable or expression. The program executes a different set of instructions depending on the value of the variable or expression. In C programming, there are three main types of conditional statements: if statements, switch statements, and ternary operators. If statements are used to execute a block of code if a specific condition is true. Switch statements are used to execute different blocks of code depending on the value of a variable. Ternary operators are used to simplify if-else statements and are often used in place of simple if-else statements.

Loops:

Loops in C programming are used to repeat a block of code a specified number of times. Loops are essential in programming as they help to execute a set of instructions repeatedly, without the need to write the same code over and over again. In C programming, there are three main types of loops: for loops, while loops, and do-while loops. For loops are used to repeat a block of code a specific number of times. While loops are used to repeat a block of code until a specific condition is false. Do-while loops are used to execute a block of code at least once and then repeat it until a specific condition is false.

Jumps:

Jumps in C programming are used to transfer control to another part of the program. Jumps are used to interrupt the normal flow of the program and transfer control to another part of the program. In C programming, there are three main types of jumps: break statements, continue statements, and goto statements. Break statements are used to exit a loop or switch statement. Continue statements are used to skip to the next iteration of a loop. Goto statements are used to transfer control to a labelled statement in the program. The use of jumps in C programming is essential for controlling the flow of the program and implementing complex control structures.

Functions:

Functions in C programming are used to group a set of statements together to perform a specific task. They help to simplify code and make it easier to read, write, and debug. Functions can be used to perform calculations, manipulate data, and execute complex algorithms. In C programming, functions have a name, return type, parameter list, and a body that contains the set of statements that perform the task.

User-defined functions:

User-defined functions in C programming are functions that are created by the programmer. They are defined by the programmer to perform a specific task that is not provided by the standard library functions. User-defined functions are an essential tool for modular programming, where large programs are broken down into smaller modules, each with its own set of functions.

Types of user-defined functions:

There are two main types of user-defined functions in C programming: void functions and value-returning functions. Void functions do not return a value and are used to perform a task or manipulate data. Value-returning functions return a value and are used to perform calculations or return data to the program. Both types of functions can have parameters, which are variables passed to the function by the calling program.

Recursion:

Recursion is a technique in C programming where a function calls itself to solve a problem. It is a powerful tool for solving complex problems and implementing algorithms. Recursion is based on the principle of divide-and-conquer, where a problem is divided into smaller sub-problems until a simple case is reached, which can be solved easily. The function then combines the solutions of the sub-problems to solve the original problem.

Examples of Recursion:

There are many examples of recursion in C programming, such as the Fibonacci sequence, the factorial of a number, and the towers of Hanoi problem. The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding numbers. The factorial of a number is the product of all the numbers from 1 to that number. The towers of Hanoi problem is a classic problem that involves moving a set of disks from one tower to another tower, following a set of rules. All of these problems can be solved using recursion in C programming.

Arrays:

Arrays in C programming are a collection of elements of the same data type. They can be used to store and manipulate data, and are an essential tool for solving many programming problems. Arrays can be one-dimensional, two-dimensional, or multi-dimensional, depending on the number of dimensions required to store the data.

1D Arrays:

One-dimensional arrays in C programming are the simplest type of array, consisting of a single row or a single column of elements. They are useful for storing lists of data, such as numbers or characters, and can be easily accessed using a loop or an index.

2D Arrays:

Two-dimensional arrays in C programming consist of multiple rows and columns of elements, forming a grid or a table. They are useful for storing and manipulating data that has a two-dimensional structure, such as images, matrices, or tables. Accessing elements in a 2D array requires two indices, one for the row and one for the column.

Multi-dimensional Arrays:

Multi-dimensional arrays in C programming consist of three or more dimensions, forming a hypercube or a tensor. They are useful for storing and manipulating data that has a complex structure, such as video frames, medical images, or scientific data. Accessing elements in a multi-dimensional array requires multiple indices, one for each dimension.

Searching and Sorting using 1D Arrays:

Searching and sorting are common operations on arrays in C programming. Searching involves finding a specific element in the array, while sorting involves rearranging the elements in the array in a specific order. There are many algorithms for searching and sorting arrays, such as linear search, binary search, bubble sort, selection sort, and quicksort.

Passing Array as Parameter to Functions:

Arrays can be passed as parameters to functions in C programming, allowing the function to manipulate the array or perform operations on its elements. When passing an array to a function, the array is actually passed by reference, meaning that any changes made to the array inside the function will affect the original array in the calling program.

Strings:

Strings in C programming are a collection of characters that represent text. They are stored as arrays of characters, with a null character '\0' at the end to indicate the end of the string. Strings are used extensively in input/output operations, text processing, and data manipulation.

String Operations:

String operations in C programming involve manipulating strings to perform various tasks, such as concatenating, comparing, copying, and searching. These operations can be performed using built-in string functions or user-defined functions.

String Operations using User-defined Functions:

User-defined string functions in C programming are functions that are created by the programmer to perform specific string operations that are not provided by the standard library functions. User-defined functions can be used to perform complex string operations or to customize the behavior of existing functions.

Built-in String Functions:

C programming provides a set of built-in string functions that can be used to perform common string operations. These functions include functions for copying strings (strcpy), concatenating strings (strcat), comparing strings (strcmp), and finding the length of a string (strlen). Other built-in functions can be used to search for substrings, convert strings to numbers, and format strings for output.

Pointers:

Pointers in C programming are variables that store memory addresses. They allow the programmer to directly manipulate memory and access data at specific memory locations. Pointers are often used for dynamic memory allocation, passing parameters to functions, and creating data structures like linked lists and trees.

Pointers and Arrays:

Arrays in C programming are a collection of variables of the same data type, stored in contiguous memory locations. Pointers can be used to access individual elements of an array by pointing to the memory address of the first element. They can also be used to manipulate arrays, such as sorting and searching, and to pass arrays as parameters to functions.

Memory Allocation:

Memory allocation in C programming is the process of assigning memory to variables or data structures at runtime. C provides two main memory allocation functions: malloc() and calloc(). The malloc() function allocates a block of memory of a specified size, while calloc() allocates a block of memory and initializes it to zero. The programmer is responsible for managing allocated memory, including deallocating it when it is no longer needed.

Pointers and Functions:

Pointers can be passed as parameters to functions in C programming. This allows the function to modify the value of the variable pointed to by the pointer, rather than just a copy of the variable. Pointers can also be used to return values from functions or to allocate memory dynamically within a function.

Example Programs using Pointers:

There are many examples of programs that use pointers in C programming. Some common examples include linked lists, binary trees, and sorting algorithms like bubble sort and quicksort. Linked lists are a data structure where each node points to the next node in the list. Binary trees are a type of data structure where each node has two children, a left child and a right child, and are often used for searching and sorting. Sorting algorithms use pointers to manipulate arrays and sort the elements in ascending or descending order.

Structures:

Structure in C is a user-defined data type that allows the programmer to group variables of different data types under a single name. It provides a way to organize data in a structured manner, making it easier to read, write and manipulate.

Array of Structures:

Array of structure is a concept in C programming where an array is created to store multiple instances of a structure. It allows the programmer to work with multiple sets of data in a more organized manner. For example, an array of structure can be used to store information about a group of students, where each instance of the structure represents information about a single student.

Struct and Pointers:

Pointers and structure in C are commonly used together to access and manipulate the members of a structure. Pointers can be used to dynamically allocate memory for a structure and to access individual members of a structure.

Union:

Union is another user-defined data type in C that allows the programmer to define a new data type that contains variables of different data types. Unlike structure, only one member of a union can be accessed at a time, which makes it useful for memory optimization.

Struct and functions:

Struct and functions is another important concept in C programming, where structures can be passed as arguments to functions. This allows the programmer to perform complex operations on structures using functions, making the code more modular and easier to maintain.

Example programs using structures:

Example programs using structures can include applications such as a student database system, where structures are used to store information about each student such as name, roll number, and grades. Another example could be a program that uses structures to store employee information such as name, ID number, and salary.

Introduction to C++:

C++ is a high-level programming language that is widely used in software development. It was developed by Bjarne Stroustrup in the 1980s as an extension of the C programming language. C++ is an object-oriented language that supports features such as encapsulation, inheritance, and polymorphism. It is used to develop a wide range of software applications, including operating systems, device drivers, video games, and scientific simulations.

Data Types:

Data types in C++ are used to specify the type of data that a variable can hold. C++ supports several built-in data types, including integer, floating-point, character, boolean, and void. These data types can be combined to create more complex data structures such as arrays, structures, and classes. C++ also provides support for user-defined data types through the use of classes.

Basic Input and Output:

Basic input and output in C++ is done using the standard input/output streams: cin and cout. The cin stream is used to read data from the keyboard, while the cout stream is used to display data on the screen. C++ also supports formatted input/output, which allows the programmer to control the format of the data that is input or output.

Operators:

Operators in C++ are used to perform various operations on variables and data. C++ supports several types of operators, including arithmetic, relational, logical, bitwise, and assignment operators. Operators can be used with built-in data types and user-defined data types to perform complex operations.

Conditional Statements:

Conditional statements in C++ are used to execute code based on a certain condition. C++ supports several types of conditional statements, including the if statement, the if-else statement, the switch statement, and the ternary operator. These statements allow the programmer to create complex control structures that can handle a variety of situations.

Loops:

Loops in C++ are used to repeat a block of code a certain number of times. C++ supports several types of loops, including the for loop, the while loop, and the do-while loop. Loops can be used to perform iterative operations, such as counting or searching, and can be combined with conditional statements to create complex control structures.

C++ Functions:

Functions in C++ are blocks of code that can be called to perform a specific task. Functions are used to simplify code and make it easier to read, write, and debug. In C++, functions can be defined with a return type, a function name, a parameter list, and a body that contains the set of statements that perform the task. Functions can be called from other functions or from the main() function.

Function Overloading:

Function overloading is a feature in C++ that allows multiple functions to have the same name but different parameters. The compiler determines which function to call based on the number, type, and order of the arguments passed to the function. Function overloading is used to provide multiple ways of calling a function with different argument lists.

Default Arguments:

Default arguments are used in C++ to provide a default value for a function parameter. If an argument is not passed when the function is called, the default value is used instead. Default arguments are useful for simplifying code and making it more readable by eliminating the need for extra parameters in function calls.

Storage Class:

Storage class in C++ is used to define the lifetime and scope of variables. There are four types of storage class in C++: automatic, static, register, and extern. Automatic variables have a local scope and are destroyed when the function in which they are declared exits. Static variables have a global scope and retain their value between function calls. Register variables are stored in the CPU registers for faster access. Extern variables are used to refer to variables declared in other files.

Recursion:

Recursion is a technique in C++ where a function calls itself to solve a problem. It is a powerful tool for solving complex problems and implementing algorithms. Recursion is based on the principle of divide-and-conquer, where a problem is divided into smaller sub-problems until a simple case is reached, which can be solved easily. The function then combines the solutions of the sub-problems to solve the original problem.

Return Reference:

Return reference is a feature in C++ that allows a function to return a reference to a variable. This is useful for functions that modify the value of a variable and return it. The returned reference can be used to modify the value of the original variable directly.

C++ Arrays and Strings:

Arrays in C++ are used to store a collection of elements of the same data type. Strings in C++ are a collection of characters. Both arrays and strings can be accessed using indexes. In C++, arrays can have one or more dimensions. Strings in C++ are represented as arrays of characters, with a null terminator character at the end.

Structure and Functions:

Structures in C++ are used to group related data together. They can be used to create complex data types that contain multiple elements of different data types. Functions can be used to manipulate and access the elements of a structure. Structures can also be passed as function arguments and returned by functions.

C++ Classes and Objects:

C++ is an object-oriented programming language, which means that it supports the creation of classes and objects. A class is a user-defined data type that encapsulates data and functions that operate on that data. An object is an instance of a class, which means it is a variable of that class type.

Constructors:

Constructors are special member functions of a class that are automatically called when an object of that class is created. They are used to initialize the data members of the object to default or user-defined values. Constructors have the same name as the class and can be overloaded to take different parameters.

Objects and functions:

Objects and functions in C++ are closely related because objects are created from classes and functions are defined within classes. Member functions are functions that operate on objects of the class, and they are defined within the class definition. Member functions can be called using the dot (.) operator on an object of the class.

Operator Overloading:

Operator overloading is a feature in C++ that allows operators to be overloaded to work with user-defined data types. This means that operators such as +, -, *, / can be used with objects of a class. Operator overloading is achieved by defining a function with the same name as the operator and the class that the operator works on.

C++ Inheritance:

Inheritance is one of the key concepts of object-oriented programming that allows a class to inherit properties and behaviors from another class. In C++, inheritance is implemented using the 'class' keyword followed by a colon and the keyword 'public' or 'private', followed by the name of the base class. The derived class can access all public and protected members of the base class, but cannot access private members.

Function Overriding:

Function overriding is a technique in C++ inheritance where a function with the same name and signature as a base class function is defined in the derived class. When an object of the derived class is created, the derived class function is called instead of the base class function. Function overriding is used to implement polymorphism in C++, where objects of different classes can be treated as objects of a common base class.

Friend Functions:

A friend function is a function that is not a member of a class, but has access to the private and protected members of the class. In C++, friend functions are declared inside the class using the 'friend' keyword. Friend functions are often used to simplify code or to improve performance by avoiding excessive copying of objects.

Chapter 3

INTRODUCTION

Part A:

The purpose of this report is to provide an overview of our project that is a stock market simulator, here we allow to user to make their profile and run a simulation of performance of their purchased stocks over the next year. It provides the user with a predicted profit or loss incurred by taking input from the user

The project was undertaken as part of our curriculum. The system was designed to use various features of the C language like loops, structures, functions, conditional statements, etc. and hence get comfortable in their use.

In this report, we will discuss the requirements, design, implementation, and testing of our stock market simulator.

Part B:

Our C++ code swaps two integer values using pointers and dynamic memory allocation. It allocates memory for two integer pointers, a and b, and sets their values to 29 and 50, respectively. It then swaps the values of a and b by adding them and storing the result in a, subtracting b's original value from the new value of a and storing the result in b, and then subtracting b's original value from the new value of a and storing the result in a. The new values of a and b are then printed, and the memory allocated for the pointers is released using the "delete" operator. This code demonstrates a simple way to swap values using pointers and dynamic memory allocation in C++.

Chapter 4

TECHNOLOGY

4.1 C Programming:

C programming is a procedural programming language that was originally designed for system programming. It is a low-level programming language that provides direct access to the computer's memory and system resources. C programming is widely used for developing operating systems, embedded systems, device drivers, and other low-level software.

C programming is known for its simplicity and efficiency, making it a popular choice for software developers. It has a relatively small set of keywords and syntax rules, which makes it easy to learn and use. C programming also provides a high degree of control over the system's hardware and resources, which is important in developing low-level software.

4.2 C++:

C++ is a high-level programming language that is an extension of the C programming language. It is an object-oriented programming language that provides additional features such as classes, inheritance, polymorphism, and templates. C++ programming is widely used for developing system software, desktop applications, video games, and other software.

C++ programming provides a powerful set of features that enable developers to write efficient and complex software. It is also known for its performance and memory management capabilities, which are important in developing software that requires high performance and low latency.

`

Chapter 5

PROBLEM STATEMENT

1. A program using C concepts to create a user-friendly bot that helps the user to predict the performance of their purchased stocks over a period.
2. Write a C++ program to swap the values of two dynamically allocated variables and release the memory after swapping. (use new & delete operators)

Chapter 6

SOFTWARE REQUIREMENT SPECIFICATIONS

- Operating System: C and C++ can be run on a variety of operating systems, including Windows, macOS, and Linux.
- Development Environment: A compiler and development environment are required to write, compile, and debug C and C++ programs. Popular development environments include Visual Studio, Code::Blocks, Eclipse, etc.
- Libraries: Some C and C++ programs may require additional libraries or frameworks, depending on their functionality.

Chapter 7

DESIGN AND IMPLEMENTATION

C Program:

This C code is a program for a simple stock market simulator that allows the user to enter the number of stocks they have purchased, and the select the stock codes. The program displays a list of stocks available for purchase on the National Stock Exchange and asks the user to enter the codes for their purchased stocks.

The program defines two structures: stocks and User. The “stocks” structure has three variables: “Stock_name”, which is an array of 20 characters to store the name of the stock; “amt”, which is an integer variable used to store the initial value of the stock; and “no_of_shares”, which is an integer variable used to store the number of shares purchased. The User structure has three variables: “name”, which is an array of 40 characters to store the user's name; “age”, which is an integer variable to store the user's age; and “email”, which is an array of 40 characters to store the user's email address.

The program has three functions: “create_user_profile()”, “check()”, and “market_sim()”. The create_user_profile() function prompts the user to enter their name, age, and email address, and stores the information in the User structure. The check() function is used to validate whether the stock code entered by the user is valid or not. If the code is not valid, the function displays an error message and exits the program. The market_sim() function is the main function that drives the program. It displays the list of stocks available for purchase, prompts the user to enter the number of stocks they have purchased, and the stock codes for each stock. The function then calculates and displays the initial value, current value, profit/loss, and percent change for each stock, as well as the total initial value, total current value, profit/loss, and percent change for all stocks combined.

The “stdlib.h” library contains the rand() function that generates random integers between 0 and the maximum value defined. The time.h library contains the time() function that is used to seed the random number generator with a value based on the current time, so that the sequence of random numbers generated is different every time the program is run.

In the code, the srand() function from the stdlib.h library is called with the argument time(NULL) to seed the random number generator with the current time, and then the rand() function is used to generate random numbers between 1 and 100 for simulating the stock prices.

The code is as follows:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#define MAX_NAME_LENGTH 40
int total_initial_value = 0, current_price = 0, total_initial_value, total_current_value, profit_loss;
float percent_change;
struct stocks
{
   char Stock_name[20];
   int amt;
   int no_of_shares;
} stock_details[10];
struct User
{
   char name[MAX_NAME_LENGTH];
   int age;
   char email[MAX_NAME_LENGTH];
};

void create_user_profile(struct User *user)
{
   printf("\e[1;34mPlease enter your name:\e[0m ");
   scanf("%s", user->name);
printf("\e[1;34mPlease enter your age:\e[0m ");
```

```
    scanf("%d", &user->age);
    getchar(); // consume newline character

    printf("\e[1;34mPlease enter your email address:\e[0m ");
    fgets(user->email, MAX_NAME_LENGTH, stdin);
    user->email[strcspn(user->email, "\n")] = '\0'; // remove newline character from email
}

void check(char a[20])
{
    // an array of strings
    int y = 0;
    char NSE[15][13] = {"JSWSTEEL", "ONGC", "TATASTEEL", "SBIN", "TECHM",
"HDFCBANK", "TCS", "ASIANPAINT", "WIPRO", "ITC", "RELIANCE", "LT",
"AXISBANK", "APOLLOHOSP", "HEROMOTOCO"};
    for (int i = 0; i < 12; i++)
    {
        if (strcmp(a, NSE[i]) == 0)
        {
            y += 1;
        }
    }
    if (y == 0)
    {
        printf("enter a valid stock code");
        exit(0);
    }
}

void market_sim()
{
    int num;
```

```
int total_initial_value = 0; // Initialize the variable to zero
    // Display the list of stocks
    printf("\t \033[0;36m-----------------------------------------------\033[0m\n");
    printf("\t \033[0;36m|\t\t National Stock Exchange          |\033[0m\n");
    printf("\t \033[0;36m-----------------------------------------------\033[0m\n");
    printf("\t \033[0;31m| Stock Code  |   Company Name                |\033[0m\n");
    printf("\t \033[0;36m-----------------------------------------------\033[0m\n");
    printf("\t \033[0;32m| JSWSTEEL    |   JSW Steel                   |\033[0m\n");
    printf("\t \033[0;32m| ONGC        |   Oil and Natural Gas Corporation|\033[0m\n");
    printf("\t \033[0;32m| TATASTEEL   |   Tata Steel Ltd.             |\033[0m\n");
    printf("\t \033[0;32m| SBIN        |   State Bank of India         |\033[0m\n");
    printf("\t \033[0;32m| TECHM       |   Tech Mahindra               |\033[0m\n");
    printf("\t \033[0;32m| HDFCBANK    |   HDFC Bank Ltd.              |\033[0m\n");
    printf("\t \033[0;32m| TCS         |   Tata Consultancy Services   |\033[0m\n");
    printf("\t \033[0;32m| ASIANPAINT  |   Asian Paints                |\033[0m\n");
    printf("\t \033[0;32m| WIPRO       |   Wipro Ltd.                  |\033[0m\n");
    printf("\t \033[0;32m| ITC         |   India Tobacco Company Ltd.  |\033[0m\n");
    printf("\t \033[0;32m| RELIANCE    |   Reliance Industries Ltd.    |\033[0m\n");
    printf("\t \033[0;32m| LT          |   Larsen & Toubro             |\033[0m\n");
    printf("\t \033[0;32m| AXISBANK    |   Axis Bank Ltd.              |\033[0m\n");
    printf("\t \033[0;32m| APOLLOHOSP  |  Apollo Hospitals Enterprise Ltd.|\033[0m\n");
    printf("\t \033[0;32m| HEROMOTOCO  |   Hero MotoCorp Ltd.          |\033[0m\n");
    printf("\t \033[0;36m-----------------------------------------------\033[0m\n");

    printf("\e[1;33mEnter the number of stocks u have purchased:  \e[0m");
    scanf("%d", &num);
    printf("\e[1;33mPlease tell us which of these stocks you have purchased using the stock codes\e[0m\n");

    // Iterate through the stocks purchased by the user
    for (int i = 0; i < num; i++)
```

```
    {
        printf("\e[1;33mEnter the stock you have purchased (code) \t \e[0m");
        scanf("%s", stock_details[i].Stock_name);
        check(stock_details[i].Stock_name);

        // Get the initial value of share and the number of shares purchased
        printf("\e[1;33mEnter the amount invested per share: \t\e[0m");
        scanf("%d", &stock_details[i].amt);
        printf("\n\e[1;33mEnter no of shares you have purchased\t\e[0m");
        scanf("%d", &stock_details[i].no_of_shares);

        // Calculate total initial value of stock
        total_initial_value += (stock_details[i].amt * stock_details[i].no_of_shares);
        percent_change = ((float)rand() / (float)(RAND_MAX)) * 0.2 - 0.1; // random change between -10% and +10%
        current_price += stock_details[i].amt * (1 + percent_change);

        // Calculate total current value of stock and profit/loss
        total_current_value = current_price * stock_details[i].no_of_shares;
    }

    printf("\n\e[1;33mNow let's predict the market performance of your stocks\e[0m\n");

    profit_loss = total_current_value - total_initial_value;

    // Output results
    printf("\e[1;32mInitial value: $%d\e[0m\n", total_initial_value);
    printf("\e[1;32mCurrent value: $%d\e[0m\n", total_current_value);
    printf("\e[1;32mProfit/Loss that you will incur in the period of 365 days: $%d\e[0m\n", profit_loss);
    printf("\e[1;33mThank you for using our System\e[0m\n");
```

```
    exit(0);
}

int main()
{
    struct User user;
    int ch;
    srand(time(NULL)); // initialize random seed
    printf("\033[1;34mHello and welcome to our stock market simulation\033[0m\n");
    while (1)
    {
        printf("\033[1;33mWhat would you like to do?\n1.Create your profile\n2.Perform Market
Simulation\n3.Exit\n\033[0m");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            create_user_profile(&user);
            printf("\033[1;32mUser profile created:\n");
            printf("Name: %s\n", user.name);
            printf("Age: %d\n", user.age);
            printf("Email: %s\n\033[0m", user.email);
            break;
        case 2:
            market_sim();
            break;
        case 3:
            exit(0);
        default:
            printf("\033[1;31mINVALID INPUT\n\033[0m");
            break;
        }
```

```
    }
    return 0;
}
```

C++ Program:

This C++ code demonstrates the exchange of two integer values using dynamic memory allocation. The program begins by printing "Before editing" on the console. It then declares two integer pointers a and b and initializes them to NULL. The new(nothrow) int syntax is used to allocate memory for integer values for both a and b. The nothrow parameter is used to avoid throwing an exception if memory allocation fails. Instead, it returns a null pointer that can be checked in subsequent code using the if(!a) and if(!b) statements. If memory allocation is successful for both pointers, the values of a and b are set to 29 and 50, respectively, and their values are printed to the console. Then, the values of a and b are exchanged by a temporary variable. The values of a and b are added to *a, and then the value of *b is obtained by subtracting the first value of *b from the new value of *a. Finally, the initial value of *a is obtained by subtracting the new value of *b from the sum.

The code is as follows:

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

int main()
{
    cout << "Before swapping." << endl;
    int *a = NULL;
    a = new (nothrow) int;
    if (!a)
        cout << "allocation of memory failed\n";
    else
    {
```

```
        *a = 29;
        cout << "Value of a= " << *a << endl;
    }
    int *b = NULL;
    b = new (nothrow) int;
    if (!b)
        cout << "allocation of memory failed\n";
    else
    {

        *b = 50;
        cout << "Value of b= " << *b << endl;
    }
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
    cout << "\nAfter swapping." << endl;
    cout << "value of a = " << *a << endl
         << "value of b = " << *b << endl;
    delete a;
    delete b;
    return 0;
}
```

Chapter 8

RESULTS

The output of the C program is as follows:

```
Hello and welcome to our stock market simulation
What would you like to do?
1.Create your profile
2.Perform Market Simulation
3.Exit
1
Please enter your name: Khushi
Please enter your age: 20
Please enter your email address: khushi@gmail.com
User profile created:
Name: Khushi
Age: 20
Email: khushi@gmail.com
What would you like to do?
1.Create your profile
2.Perform Market Simulation
3.Exit
2
```

Fig 8.1 – Output Slide 1

```
2
        ------------------------------------------------
        |              National Stock Exchange         |
        ------------------------------------------------
        | Stock Code  |   Company Name                 |
        ------------------------------------------------
        | JSWSTEEL    |   JSW Steel                    |
        | ONGC        |   Oil and Natural Gas Corporation|
        | TATASTEEL   |   Tata Steel Ltd.              |
        | SBIN        |   State Bank of India          |
        | TECHM       |   Tech Mahindra                |
        | HDFCBANK    |   HDFC Bank Ltd.               |
        | TCS         |   Tata Consultancy Services    |
        | ASIANPAINT  |   Asian Paints                 |
        | WIPRO       |   Wipro Ltd.                   |
        | ITC         |   India Tobacco Company Ltd.   |
        | RELIANCE    |   Reliance Industries Ltd.     |
        | LT          |   Larsen & Toubro              |
        | AXISBANK    |   Axis Bank Ltd.               |
        | APOLLOHOSP  |  Apollo Hospitals Enterprise Ltd.|
        | HEROMOTOCO  |   Hero MotoCorp Ltd.           |
        ------------------------------------------------
Enter the number of stocks u have purchased:  3
Please tell us which of these stocks you have purchased using the stock codes
Enter the stock you have purchased (code)        LT
Enter the amount invested per share:    2000

Enter no of shares you have purchased   100
Enter the stock you have purchased (code)        ITC
Enter the amount invested per share:    1000

Enter no of shares you have purchased   50
Enter the stock you have purchased (code)        ONGC
Enter the amount invested per share:    500

Enter no of shares you have purchased   1000

Now let's predict the market performance of your stocks
Initial value: $750000
Current value: $3538000
Profit/Loss that you will incur in the period of 365 days: $2788000
```

Fig 8.2 – Output Slide 2

The output of the C++ program is as follows:

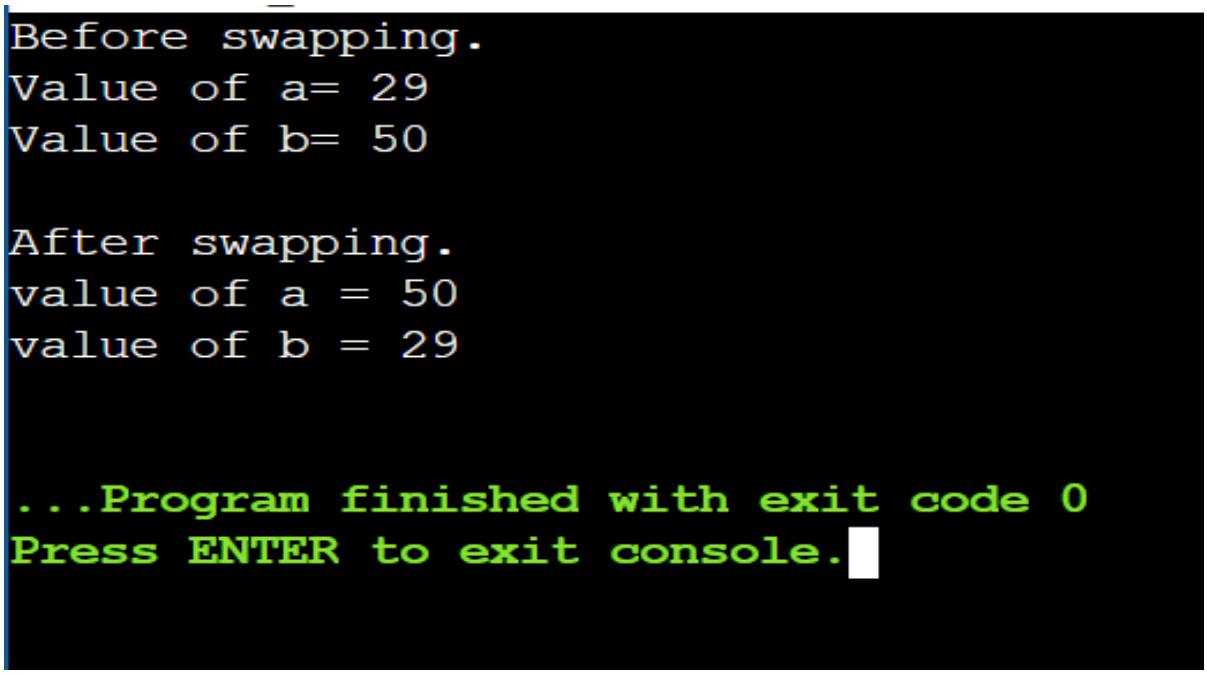

Fig 8.3 – Output Slide 3

CONCLUSION

In conclusion, this C and C++ internship provided a great opportunity for us to gain hands-on experience in these programming languages. During the internship, we learnt various concepts such as data types, control statements, functions, arrays, pointers, dynamic memory allocation, and other advanced features of C and C++.

As a part of this internship, we worked on a project that simulated a stock market using C programming language. The project helped me to understand the concepts of data structures, file handling, and algorithms. By developing this project, I gained valuable knowledge about the financial domain, and how to handle and process large datasets efficiently.

In addition to this, we also worked on a project using C++ that involved dynamic swapping of numbers using pointers and dynamic memory allocation. This project helped me to understand the basics of object-oriented programming, and how to use C++ to implement more complex algorithms.

Overall, this internship has been a great learning experience for us. It has not only helped us to improve our technical skills, but also provided me with an opportunity to work in a team, collaborate on a project, and gain real-world experience in the field of computer programming.

BIBILIOGRAPHY

[1] H. Schildt, C++: The Complete Reference, 4th ed. New York, NY: McGraw-Hill, 2002.

[2] B. W. Kernighan and D. M. Ritchie, The C Programming Language, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[3] M. D. Diklic, "Object-oriented programming in C," IEEE Software, vol. 10, no. 3, pp. 25-34, May 1993.

[4] J. R. Wilson, "C++ as a parallel programming language," Parallel Computing, vol. 19, no. 8, pp. 927-946, 1993.

[5] M. J. Morris, "An empirical comparison of C, C++, and Ada," in Proceedings of the 1991 ACM SIGPLAN Conference on Programming Language Design and Implementation, New York, NY, 1991, pp. 244-253.

[6] A. T. Burrell, "Object-oriented programming in C++ for scientific computing," in Proceedings of the 1992 ACM/IEEE Conference on Supercomputing, Washington, DC, 1992, pp. 154-161.

7] The C++ Programming Language, http://www.stroustrup.com/C++.html, accessed on Oct. 22, 2022.

[8] C Programming.com, http://www.cprogramming.com, accessed on Oct. 20, 2022.