INTERNSHIP REPORT

A report submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

Ajay Kumar Yadav

Regd. No. : CSJMA 24001390012

Summer Internship Training

at CODEALPHA

(Duration : 10/08/2025 to 10/09/2025)

Computer Science & Engineering

University Institute of Engineering and Technology, CSJMU

(A State Government University)

KANPUR, UTTAR PRADESH , 208024

2024-2028

DECLARATION BY CANDIDATE

I hereby declare that the work presented in this report entitled “C++ programming during internship at Code Alpha” in the fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering, University Institute Of Engineering and Technology, C.S.J.M. University Kanpur, is an authentic record of my own work at CodeAlpha carried out over a period from 10/08/2025 to 10/09/2025.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Ajay Kumar Yadav

CSJMA24001390012

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Team CodeAlpha

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to CodeAlpha for providing me with the opportunity to undertake this enriching summer internship on C++Programming .This experience has greatly enhanced my technical skills and deepened my understanding of Programming Language.

I extend my heartfelt thanks to my mentors and the entire team at CodeAlpha for their guidance, support, and constructive feedback throughout the internship. Their expertise and encouragement have been invaluable in improving my problem-solving abilities and practical knowledge.

I would also like to acknowledge my fellow interns for their camaraderie and collaboration, which made this learning experience even more fulfilling.

Lastly, I am grateful to my family and friends for their continuous support and motivation during this internship journey.

Thank you all for helping me grow professionally and personally.

Sincerely,

Ajay Kumar Yadav

CSJMA24001390012

CERTIFICATE

CONTENT

PROJECT OVERVIEW

1.Number guessing game

2.Simple Calculator

3.To do List

What is programming language?

In computer science, a programming language is a formal set of instructions used to produce various kinds of output. It allows programmers to create sequences of instructions that a computer can understand and execute. Here's a detailed breakdown:

Syntax and Semantics:

Syntax: The rules and structure of the language. It dictates how symbols, keywords, and punctuation are arranged to form valid code. Syntax errors occur when code doesn't adhere to these rules.

Semantics: The meaning behind the valid code. It defines the behavior of the instructions. Even if code is syntactically correct, incorrect semantics can lead to logical errors.

Types of Languages:

High-level Languages: Designed to be human-readable and easier to understand. Examples include Python, Java, C++, etc.

- **Low-level Languages:** Closer to machine code and hardware. They are less human-readable but offer more direct control over hardware. Assembly language is an example.

Features of Programming Languages:

- **Variables and Data Types:** Languages have different data types (integers, strings, Booleans) and methods to store and manipulate them.

- **Control Structures:** Loops, conditionals, and branching statements control the flow of execution in code.

- **Functions and Methods:** Blocks of code that can be reused to perform specific tasks, promoting modularity and reusability.

- **Libraries and Frameworks:** Pre-wri-en code that provides additional functionality and tools to simplify development.

- **Paradigms:** Different languages follow different paradigms, such as procedural, object-oriented, functional, or logic programming.

Translation Process:

- **Compiler:** Translates entire code into machine code before execution. For example, C++.

- **Interpreter:** Translates code line by line and executes it immediately. Examples include Python and JavaScript.

- **Hybrid Approaches:** Some languages use a mix of compilation and interpretation.

Application Areas:

C++ is used across various domains, including:

- **System Development:** C++ is widely used for building robust system software (e.g., operating systems and device drivers). **Game Development:** C++ is the primary
- language for high-performance game engines and 3D graphics.

- **Enterprise Applications:** C++ is used in enterprise-level applications requiring high efficiency and scalability (e.g., trading systems).
- **Embedded Systems:** Many embedded devices utilize C++ for its object-oriented nature and speed.

Introduction to C++ Programming:

C++ is a powerful language used in system/software development, game development, embedded systems, and competitive programming. C++ programs typically have a main() function where execution starts.

Features of C++:

- **High-Level & Object-Oriented:** Easy to read and supports OOP concepts like classes, inheritance, and polymorphism.

- **Low-Level Manipulation:** Supports pointers and direct memory access like C

- **Powerful and Efficient:** Offers manual memory management, strong performance, and fine control over system resources..

- **Rich Standard Library:** Provides wide range of functions, containers and algorithm(stl)etc.

Basic Structure:

- C++ programs are built using functions and classes, with execution starting from the main() method. Supports various data types, control structures (if-else, loops), and
- provide manual memory management using new and delete.

Applications:

- Used in System development, Game development, Enterprise Application, and Embedded systems.

Variables and Data Types in C++:

In C++ programming, variables and data types are essential components used to store and manipulate data. Here's an in-depth look at variables and various data types in C++:

Variables:

A variable in C++ is a container that holds data during the execution of a program. Each variable has a data type that defines the kind of value it can hold.

Syntax for Variable Declaration:

data_type variable_name;

Rules for Naming Variables:

- Must begin with a le-er (uppercase or lowercase) or a dollar sign ($) or an underscore (_).
- Subsequent characters can be letters, digits, dollar signs, or underscores.
- C++ is case-sensitive, so var and Var are different variables.
- Variables cannot use C++ reserved keywords.

Data Types:

C++ provides several fundamental data types to represent different kinds of data. Here are the primary data types:

Primitive Data Types:

Integer Types:

- **int:**
  - **Size**: 32 bits (4 bytes)
  - **Description**:Themostcommonlyusedinteger data type. It can represent values from -2,147,483,648 to 2,147,483,647.
  - **Declaration and Initialization**:

    int i = 50000; // Declaration and initialization

- **Short:-**
  - **Size**: 16 bits (2 bytes)
  - **Description**:Usefulforsavingmemoryinlargearrayswhenthememory savings actually matters. It can store values from -32,768 to 32,767.
  - **Declaration and Initialization :**

short s=١٠٠٠٠; //Declaration and initialization

- **long:**

- **Size**: 64 bits (8 bytes)
- **Description**: Used when a wider range than int is needed. It can store values from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
- **Declaration and Initialization:**

```
long l = 100000L; // Declaration and initialization
```

Floating-Point Types:

- **float:**
  - **Size**: 32 bits (4 bytes)
  - **Description**: Used for single-precision floating-point numbers. It has a precision of about 7 decimal digits.
  - **Declaration and Initialization :**

```
float f = 10.5f; // Declaration and initialization
```

- **double:**
  - **Size**: 64 bits (8 bytes)
  - **Description:** Used for double-precision floating-point numbers. It has a precision of about 15 decimal digits.

- **Declaration and Initialization :**

double d = 20.5; // Declaration and initialization

- **Character Types:**
  - **char:**
    - **Size**: 8 bits (1 bytes)
    - **Description**: Represents a single 8-bit ASCII character. It can store any character in the ASCII set.
    - **Declaration and Initialization**: char c = 'A'; // Declaration and initialization

Boolean Types:

- **boolean:**
  - **Size**: 1 bit (not precisely defined in Java but typically 1 byte in memory)
  - **Description**: Represents one of two values: true or false. Used for conditional statements.
  - **Declaration and Initialization**:

boolean flag = true; // Declaration and initialization

Reference Data Types:

- **Arrays:**
    - A container that holds a fixed number of values of a single type. The lengthofan arrayisestablishedwhen it is created.
    - **Declaration and Initialization**:

Int arr[] = {1, 2, 3, 4, 5}; // Declaration and initialization

- **Strings:**
    - **Description**: A sequence of characters. Strings are mutable +I+n, C meaning once created, they can be altered.
    - **Declaration and Initialization**:

string str = "Hello, World!"; // Declaration and initialization

    - **Classes:** Define custom objects that can hold multiple fields and methods.

- **Interfaces:** Define a contract that classes can implement.

- **Enums**:Special classes that represent fixed sets of constants.

Control structure in C++ :

1. Conditional Statements in C++

- **if Statement :**

```
int num = 10;

if (num > 0) {

cout << "Number is positive";
} else if (num == 0) {
cout <<"Number is zero";
} else {
cout << "Number is negative";
}
```

- **switch Statement** int choice = 2; switch (choice) {

```
case 1:

cout << "Option 1 selected";
break;

case 2:

cout << “Option 2 selected”;
break;
default:
cout << “Option 2 selected”;

}
```

2. Looping Statements:

- for Loop

```
for (int i = 0; i < 5; i++) {
cout << I << “ “;
}
```

- while Loop

```
int j = 0; while (j < 5) {
```

```
cout << j << “ ”;

j++;

}
```

- do-while Loop

```
int k = 0;

do {

cout << k << " ";

k++;

} while (k < 5);
```

1. **Control Flow Statements:**

- break Statement

```
for (int i = 0; i < 10; i++) {
```

```
if (i == 5) {

break; // Exit the loop when i reaches 5

}

cout << i << “ ”;

}
```

- continue Statement

```
for (int i = 0; i < 5; i++) {

if (i == 2) {

continue; // Skip the current iteration when i equals 2

}

cout << i << " ";

}
```

1. **Nested Control Structures:**

- This can refer to combining loops, conditional statements, etc., within each other

Function:

In C++, a function, also known as a method, is a block of code that performs a specific task.

Method Declaration and Usage:

- Function declaration int add(int a, int b) {

return a + b;// Return the sum of 'a' and 'b'

}

Example:

```
Int main() {
int result = add(5, 3); // Function call
 cout << result ; // Print the result
}
}
```

In this C++ example:

- add is a function defined to take two integer parameters (int a and int b) and return their sum.

- The main function calls the add function and prints the sum using cout.

Explanation:

- **add:** The function name.
- **int:** The return type, indicating the function returns an integer.
- **int a, int b:** Parameters of the function.
- **return a + b:** Returns the sum of the two integers passed as arguments.

Array in C++:

In C++ Arrays are stored in contiguous memory locations and allow storing multiple values of the same type.

Syntax for Array Declaration:

Data_type array_name[array_size];

- **data_type**: Type of elements in the array (integers, characters, etc.).
- **array_name:** Name of the array.

- **array_size:** Number of elements the array can hold.

Example:

```
#include <iostream.h>
#include <array>

int main() {

// Declaration and initialization of an integer array
int numbers[] = {10, 20, 30, 40, 50}; //Array initialization
cout << "Array elements: ";
```

- Accessing and printing array elements using a for loop

```
for (int i = 0; i < numbers.size(); i++) {

cout << numbers[i] << “ “;

}
```

Explanation:

- int numbers[] declares an array named numbers capable of holding integers.
- {10, 20, 30, 40, 50} initializes the array with these values.
- The for loop is used to access and print each element of the array.

In this example:

- An integer array named numbers is declared with a size of 5.
- Values are assigned to each element of the array.
- The for loop iterates through the array to print each element.

Important Points:

- In C++, the size of the array is defined when it is created, and it cannot be changed a^er allocation.
- Elements in the array are accessed using indices starting from 0 to arraySize - 1.
- Arrays provide a convenient way to store and access multiple elements of the same type.

Notes:

- Arrays provide a way to store multiple values of the same type in contiguous memory.
- Indexing starts from 0. For example, numbers[0] refers to the first element, numbers[1] to the second, and so on.
- The size of the array must be known when it is created, and it cannot be changed during run time.

- Arrays can be multidimensional, allowing them to store elements in multiple rows and columns.

Projects Overview:

1. Number Guessing Game

The number guessing game is a simple application where the computer randomly selects a number within a specified range, and the player attempts to guess that number. The game provides feedback on whether the guess is too high, too low, or correct.

• Code Screenshot:

```cpp
#include<iostream>
#include<cstdlib>
#include<ctime>

using namespace std;

int main() {
    srand(time(0));
    int n;
    int max = 100;
    int min = 1;
    int to_guess = min + rand()%(max-min+1);
    while(1) {
        cout << "enter a number :";
        cin >> n;
        if(n > to_guess) cout << "too high" << endl;
        else if(n < to_guess) cout << "too low" << endl;
        else {
            cout << "you guessed it correct";
            break;
        }
    }

    return 0;
}
```

Summary of code:

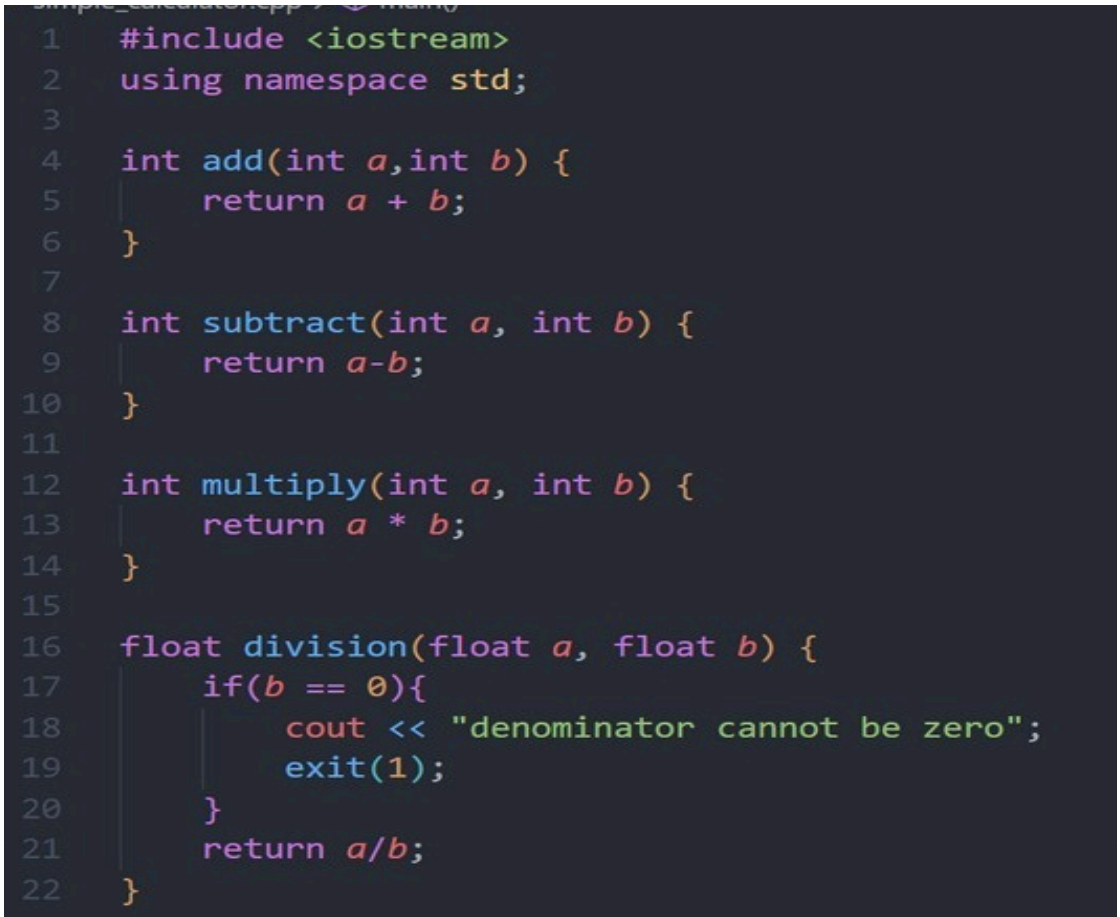
- The program generates a random number and prompts the player to guess it.
- After each guess, the player receives feedback indicating if the guess is too high or too low.
- If the player guesses correctly, they are congratulated. If not, they are allowed to guess again.
- The game loops until the player guesses the correct number , displaying you guessed it correct.

2. Simple Calculator:

A tool to calculate simple arithmetic operations like addition, subtraction

Multiplication and division.

- **Code Screenshot**



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

int add(int a,int b) {
    return a + b;
}

int subtract(int a, int b) {
    return a-b;
}

int multiply(int a, int b) {
    return a * b;
}

float division(float a, float b) {
    if(b == 0){
        cout << "denominator cannot be zero";
        exit(1);
    }
    return a/b;
}
```

```cpp
int main() {
    int a;
    int b;
    int op;
    cout << "enter a number :";
    cin >> a;
    cout << "enter a number :";
    cin >> b;

    cout << endl;

    cout << "enter 1 for addition" << endl;
    cout << "enter 2 for substraction" << endl;
    cout << "enter 3 for multiplication" << endl;
    cout << "enter 4 for division" << endl;

    cout << "enter your choice :";
    cin >> op;

    if(op == 1)  cout << add(a, b);
    else if(op == 2) cout << subtract(a, b);
    else if(op == 3) cout << multiply(a, b);
    else if(op == 4) cout << division(a, b);

    return 0;
}
```

- **Summary of code:**

- Take two numbers from the user.
- Take the desired operation (+,-,*,/).

- Use conditional statement (if else) to determine which operation to perform.
- Perform the arithmetic operation on the input numbers.
- Display the result of the operation to the user.
- Handle division by zero error

3. To do List

The project allows users to add task, view task, mark done task, remove task

User interface as user can view all the tasks , remove existing one either mark done or not. This helps to people to plan their day ahead .

Summary:

- **Data Structure**:

Uses a struct T with desc (task description) and done (status).

Stores tasks in a vector<T>.

- **Functions**:

add() – Adds a new task with status "Pending".

view() – Displays all tasks with their number and status ("Done" or "Pending").

mark() – Marks a selected task as "Done".

remove() – Removes a selected task from the list**.**

- **User Interaction** (main loop):

Displays a menu repeatedly: Add, View, Mark Done, Remove, Exit.

Performs the corresponding action based on the user’s choice.

- **Flow Example**:

User adds task "Buy groceries".

User adds task "Call friend".

Viewing shows:

1. Buy groceries [Pending]
2. Call friend [Pending]

User marks task 1 as done → status becomes [Done].

User removes task 2 → only task 1 remains.

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;

struct T {
    string desc;
    bool done;
};

vector<T> t;

void add() {
    T newT;
    cout << "Enter task: ";
    cin.ignore();
    getline(cin, newT.desc);
    newT.done = false;
    t.push_back(newT);
}

void view() {
    for (int i = 0; i < t.size(); i++)
        cout << i + 1 << ". " << t[i].desc << " [" << (t[i].done ? "Done" : "Pending") << "]\n";
}


void mark() {
    int i;
    view();
    cout << "Enter task number to mark done: ";
    cin >> i;
    if (i > 0 && i <= t.size()) t[i - 1].done = true;
}

void remove() {
    int i;
    view();
    cout << "Enter task number to remove: ";
    cin >> i;
    if (i > 0 && i <= t.size()) t.erase(t.begin() + i - 1);
}

int main() {
    int ch;
    cout << "-----To-do List-----\n";
    while (1) {
        cout << "\n1. Add\n2. View\n3. Mark Done\n4. Remove\n5. Exit\nEnter your response :";
        cin >> ch;
        if (ch == 1) add();
        else if (ch == 2) view();
        else if (ch == 3) mark();
        else if (ch == 4) remove();
        else if (ch == 5) break;
    }
    return 0;
}
```

Conclusion

During my internship with CodSoft, I gained a solid foundation in C++ programming by thoroughly exploring its core concepts and practical applications. Through this training, I not only developed a strong understanding of C++'s fundamentals but also applied these concepts to build two projects: a Number Guessing Game, Simple Calculator, To do List. These hands-on experiences allowed me to strengthen my coding skills, enhance my problem- solving abilities, and better understand the process of software development from concept to execution.

This internship has been instrumental in helping me grow as a programmer, and I am confident that the knowledge and skills I acquired will prove beneficial in my future endeavors in software development