**Week 1 & 2 Detailed Lecture Notes**

**Lecture 1**

**Introduction to the Course**

This course is designed to teach problem-solving techniques and programming using C++. We will learn how to design and analyze problems, break them down into smaller parts, and write efficient programs.

**Problem-Solving Methodology**

Problem-solving in programming involves the following steps:

1. **Understanding the Problem** - Read and analyze the given problem.
2. **Planning a Solution** - Think about how to solve the problem.
3. **Designing an Algorithm** - Write step-by-step instructions.
4. **Writing the Code** - Convert the algorithm into a programming language.
5. **Testing and Debugging** - Run the program and fix errors if any.
6. **Optimization** - Improve efficiency if required.

**Design, Analyze, and Decompose a Problem**

* **Design**: Planning how to solve the problem.
* **Analyze**: Checking the feasibility of the solution.
* **Decompose**: Breaking the problem into smaller, manageable parts.

Example: Suppose we need to make a cup of tea. The steps can be broken into:

1. Boil water.
2. Add tea leaves.
3. Add sugar and milk.
4. Stir and serve.

This is an example of decomposing a problem into smaller tasks.

**Algorithms, Pseudocode, and Flowcharts**

**Algorithm**

An algorithm is a step-by-step procedure to solve a problem. Example: Algorithm for adding two numbers:

1. Start
2. Input two numbers
3. Add the numbers
4. Display the sum
5. Stop

**Pseudocode**

Pseudocode is a way of writing an algorithm using simple, human-readable statements.

BEGIN

INPUT num1, num2

sum = num1 + num2

PRINT sum

END

**Flowchart**

A flowchart is a graphical representation of an algorithm using different symbols.

Example flowchart for adding two numbers:

[Start]

|

V

(Input two numbers)

|

V

(Add the numbers)

|

V

(Display sum)

|

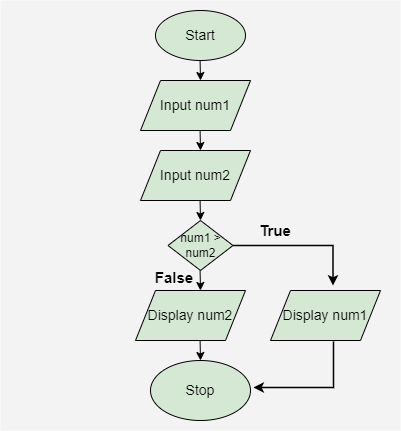
V

[Stop]

**Rules For Creating a Flowchart**

A flowchart is a graphical representation of an algorithm. It should follow some rules while creating a flowchart

* Rule 1: Flowchart opening statement must be ‘start’ keyword.
* Rule 2:  Flowchart ending statement must be ‘end’ keyword.
* Rule 3: All symbols in the flowchart must be connected with an arrow line.
* Rule 4: Each decision point should have two or more distinct outcomes.
* Rule 5: Flow should generally move from top to bottom or left to right.
* For more info check the link <https://www.geeksforgeeks.org/an-introduction-to-flowcharts/>

b

**Lecture 2**

**History of C++ Language**

* Developed by **Bjarne Stroustrup** in 1979.
* Based on C language but with additional features like **Object-Oriented Programming (OOP).**
* Widely used for **system programming, game development, and real-time applications.**

**Translators**

A translator converts high-level code into machine code. The types of translators are:

1. **Compiler**: Converts the entire code at once (e.g., C++ compiler).
2. **Interpreter**: Translates and executes the code line by line (e.g., Python interpreter).
3. **Assembler**: Converts assembly language into machine code.

**Basic Program Structure in C++**

Every C++ program consists of:

1. **Header Files** (e.g., #include <iostream>)
2. **Main Function** (int main() { })
3. **Statements** (instructions inside { })
4. **Return Statement** (return 0;)

Example:

#include <iostream>

using namespace std;

int main() {

cout << "Hello, World!";

return 0;

}

**Directives, Comments, Output**

* **Directives**: Commands that begin with # (e.g., #include <iostream>).
* **Comments**: Used for explanation (// single-line comment, /\* multi-line comment \*/).
* **Output using cout**:

cout << "Welcome to C++";

**Escape Sequences**

Escape sequences are special characters used in output.

* \n - New line
* \t - Tab space
* \" - Double quote
* \\ - Backslash

Example:

cout << "Hello\nWorld"; // Outputs: Hello (new line) World

**setw, endl Manipulator**

* **setw**: Sets the width of output.
* **endl**: Moves output to the next line.

Example:

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

cout << setw(10) << "Hello" << endl;

cout << "World";

return 0;

}

**Week 2 Lecture Notes**

**Lecture 3**

**Declaration of a Variable and Memory Concepts**

* A **variable** is a named location in memory used to store data.
* Before using a variable, it must be **declared**.
* Syntax:
* data\_type variable\_name;
* Example:
* int age;
* float temperature;

**Integer and Floating-Point Variables**

* **Integer (int)**: Stores whole numbers (e.g., 10, -5).
* **Floating-point (float, double)**: Stores decimal numbers (e.g., 3.14, -2.5).

**Initialization of Variables**

Assigning a value at the time of declaration.

int num = 10;

float pi = 3.14;

**Taking Input from User using cin**

The cin object is used to take input from the user. Example:

#include <iostream>

using namespace std;

int main() {

int age;

cout << "Enter your age: ";

cin >> age;

cout << "You entered: " << age;

return 0;

}

**Lecture 4**

**Arithmetic Operators**

C++ provides the following arithmetic operators:

* + Addition
* - Subtraction
* \* Multiplication
* / Division

**Arithmetic Expressions**

Expressions that perform arithmetic operations. Example:

int a = 10, b = 5;

int sum = a + b; // sum = 15

int difference = a - b; // difference = 5

int product = a \* b; // product = 50

int quotient = a / b; // quotient = 2

**Note:** When dividing integers, the result is also an integer. Example:

int result = 7 / 2; // result = 3 (not 3.5)

For a floating-point result, use float or double:

float result = 7.0 / 2; // result = 3.5