CSE 232

# Programming with C++

## Lecture 2

### C++ Basics


*Prepared by*

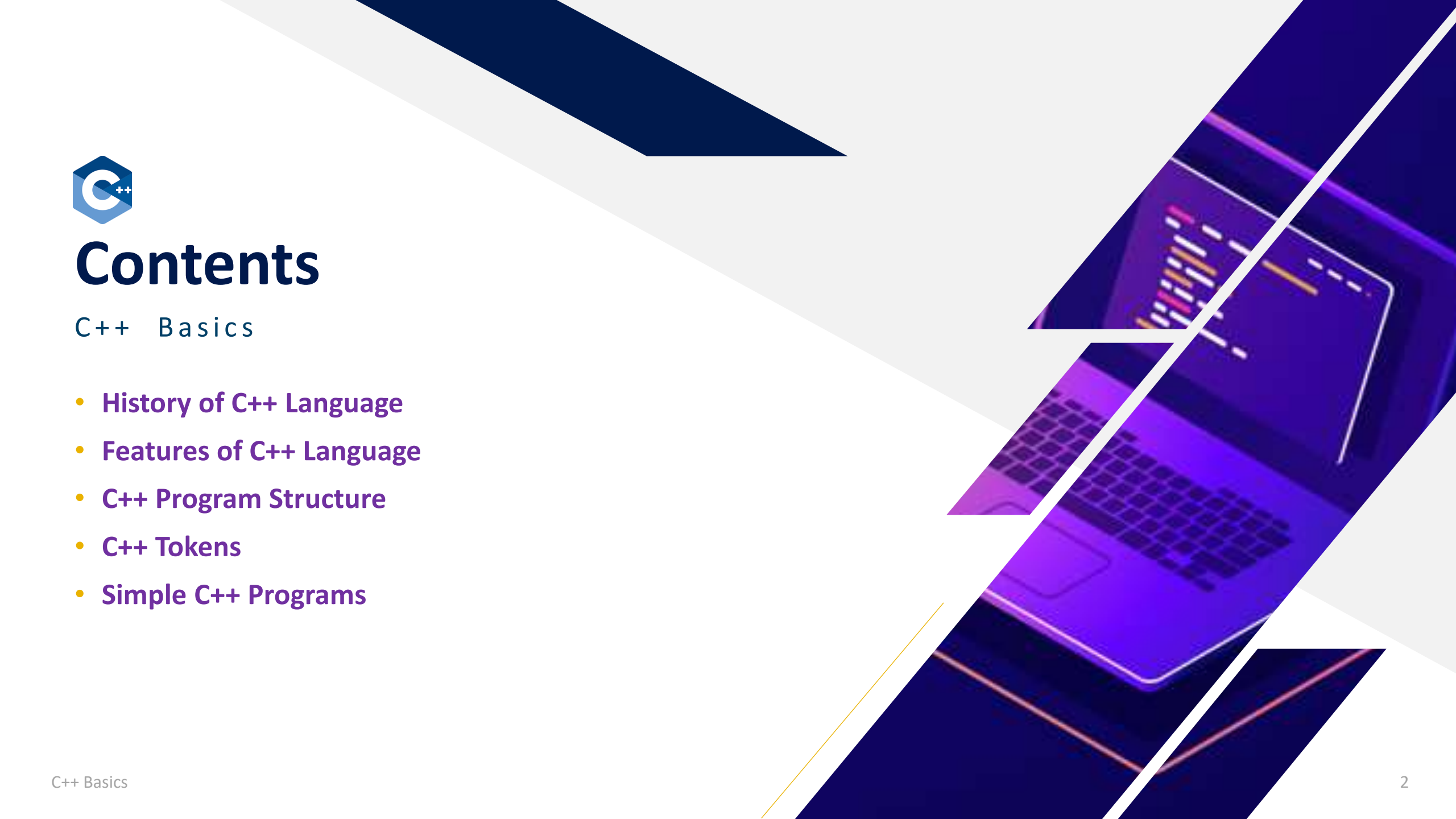
**Md. Mijanur Rahman, Prof. Dr.**

Dept. of Computer Science and Engineering

**Jatiya Kabi Kazi Nazrul Islam University, Bangladesh**

Email: mijan@jkkniu.edu.bd

Web: mijanrahman.com



# Contents

C++ Basics

- History of C++ Language
- Features of C++ Language
- C++ Program Structure
- C++ Tokens
- Simple C++ Programs

# History of C++ Language…

- C++ is a general-purpose programming language with a rich history that spans several decades. Its development and evolution have made it one of the most widely used languages in the world.
- **Origins:**
- **1979:** C++ was initially conceived by **Bjarne Stroustrup at Bell Labs** in New Jersey. Stroustrup was working on his Ph.D. thesis, which involved creating a language that combined the efficiency and flexibility of C with the advanced features of Simula, **the first object-oriented programming language**.
- **1983:** The language was originally called **"C with Classes,"** reflecting its core idea of extending the C programming language by adding classes and object-oriented features. In 1983, it was renamed C++ (where ++ signifies the increment operator in C, symbolizing an evolution of C).

# History of C++ Language…

- **Early Development:**
- **1985:** The first edition of Stroustrup's book "The C++ Programming Language" was published, marking the official release of C++ to the public. This helped popularize the language, and many developers began adopting it for software development.
- **1989:** The first major commercial implementation of C++ was released. During this period, C++ evolved with the addition of new features like function overloading, default arguments, and abstract classes.

# History of C++ Language…

- **Standardization:**
- **1990:** C++ continued to grow in popularity, and the ANSI (American National Standards Institute) committee began the formal standardization process. This was an important milestone in making C++ a more stable and widely accepted language.
- **1998:** The first international standard for C++ was published by ISO (International Organization for Standardization) as ISO/IEC 14882:1998, also known as C++98. This version standardized many features, including the Standard Template Library (STL).

# History of C++ Language…

- **Further Evolution:**
- **2003:** A minor revision, known as C++03, was released. It mainly included bug fixes and technical corrections to C++98 without introducing any major new features.
- **2011:** C++11 (also called C++0x during its development) was a major update to the language. It introduced many new features, including: Lambda expressions, Smart pointers, The auto keyword, Range-based for loops, nullptr for null pointers, etc.
- **2014:** C++14 was a relatively small update that included improvements and bug fixes to C++11, along with some new features like generic lambdas.
- **2017:** C++17 introduced additional features such as: Structured bindings, std::optional, std::variant, and std::any, Improvements to the STL.
- **2020:** C++20 brought significant new features, making it one of the most impactful updates since C++11. Notable additions included: Concepts, Coroutines, The <=> operator, Modules, Ranges, etc.

# History of C++ Language

- **Current State and Future:**
- **2023 and Beyond:** C++ continues to evolve, with plans for future versions (like C++23 and beyond). The language is widely used in system software, game development, high-performance applications, and other fields requiring efficient low-level computation.

- **Influence:**
- C++ has had a profound influence on many other programming languages, including Java, C#, and Rust. Its combination of object-oriented programming, low-level memory control, and high performance has made it a go-to language for software that requires fine control over system resources.

# Features of C++ Language…

- C++ is a general-purpose, object-oriented programming language that was created as an extension of the C programming language.
- It was designed to provide **low-level control and high-level abstractions, and it is widely used for developing high-performance, large-scale systems, as well as for creating applications in areas such as game development, scientific computing, and financial modeling.**

# Features of C++ Language…

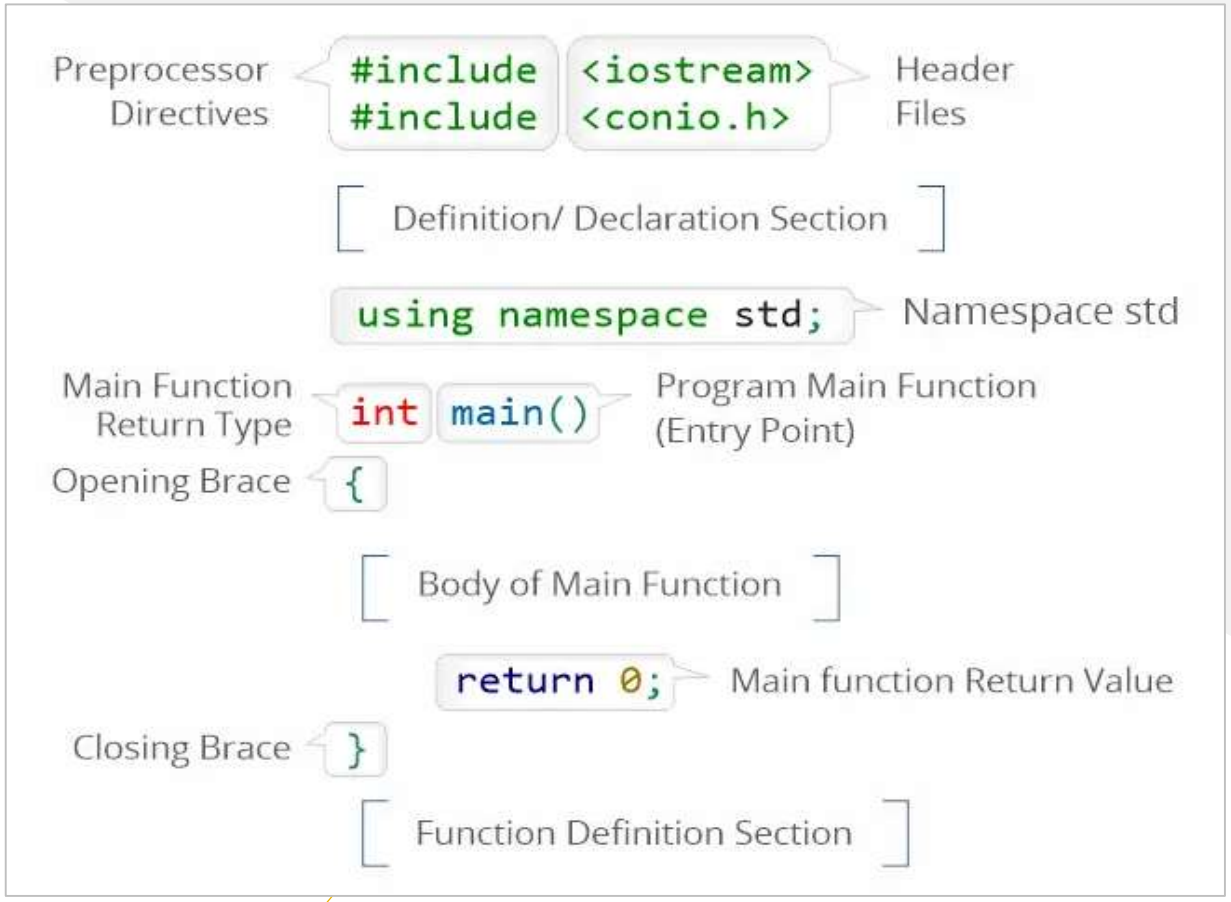
- The key features of C++ language are:
- **Object-Oriented Programming:** C++ is an object-oriented programming language, which means that it supports encapsulation, inheritance, polymorphism, and dynamic dispatch, as well as other OOP concepts such as classes and objects. This makes it possible to model complex, real-world systems and to write modular, reusable code.
- **C++ Standard Template Library (STL):** The C++ Standard Template Library (STL) is a collection of generic algorithms and data structures, such as vectors, lists, and maps, that can be used to solve a wide range of problems. The STL provides a high level of abstraction, making it easier to write code that is efficient and scalable.
- **Templates:** Templates are a powerful feature of C++ that allows for generic programming, where a single piece of code can work with objects of different types. This makes it possible to write generic algorithms that can work with any data type, which can be particularly useful for developing reusable components.

# Features of C++ Language

- **Exception Handling:** C++ provides support for exception handling, which is a mechanism for dealing with runtime errors in a structured way. This makes it easier to write robust code that can handle unexpected conditions, and it also provides a way to separate error-handling code from the main logic of a program.
- **Low-Level Control:** C++ provides low-level control over the underlying hardware, making it possible to write code that is efficient and close to the metal. This makes it a good choice for developing high-performance systems, where speed and efficiency are critical.
- **Operator Overloading:** C++ allows operators, such as +, -, *, and /, to be overloaded, which means that they can be redefined to work with user-defined types. This makes it possible to create custom types that behave like built-in types, which can make code easier to read and understand.
- These features make C++ a versatile and flexible programming language that can be used to solve a wide range of problems. Additionally, C++ has a large, active community of developers and a wealth of libraries and tools available.

# C++ Program Structure…

- Basic structure of a C++ program:



# C++ Program Structure…

- The C++ program structure is divided into several sections which are namely headers, class definitions, member functions definitions, and the main function. The basic structure of a C++ program, including:
- **Preprocessor Directives:** Preprocessor directives are instructions that are processed by the preprocessor before the actual compilation of the program. They are typically used to include header files, define macros, and control the behavior of the compiler. The preprocessor directives start with a # symbol. It is used to include the necessary header file in a C++ program before compilation.
- **Header File:** The Header File contains the function declaration and macro definition for C++ in-built library functions which we use in our C++ program during programming. The header files are included in the C++ program using the #include <filename.h> command. For example, #include <iostream.h> , #include <math.h> , etc.

# C++ Program Structure…

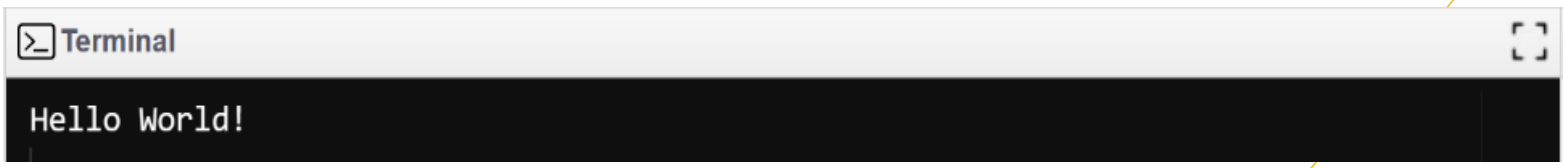
- **Namespace:** A namespace is a container for a set of identifiers that provides a way to avoid name collisions between different parts of a program. Namespaces can be used to group related classes, functions, and variables, and they are typically defined using the "namespace" keyword.
- When we use ‘using namespace std’ in a C++ program, it does not require writing std:: in front of standard commands throughout the code. Namespace std contains all the classes, objects, and functions of the standard C++ library.
- **Definition/Declaration Section:** This section is used to define macro, structure, class, and global variables to be used in the programs. These variables can be used throughout the program.
- **Program Main Function:** In C++, the main function is treated as the entry point of the program. It has a return type and a body. The default value of the return type is 0. We will write executable statements in the body. The main function starts with the Opening Brace ‘{‘, and ends with Closing Brace ‘}’ (the end of the program). The main function is called by the operating system when the user runs the program.
- **Function Definition Section:** We can define other user-defined function in this section that fulfills a particular requirement.

# C++ Program Structure

- **Hello World Program in C++:**

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

int main() {
    cout << "Hello World!" << endl;
    return 0;
}
```



# C++ Tokens…

- C++ tokens are the smallest building blocks of a C++ program.
- They are the basic elements that make up the syntax of a C++ program, and they are used to represent identifiers, keywords, constants, operators, and punctuation.

- Tokens are used to define the structure of a program, to perform operations, to declare variables and functions, and to control the flow of execution of a program.

# C++ Tokens…

- The following are some common types of C++ tokens:
- **Keywords:** Keywords are reserved words in C++ that have a special meaning, and they cannot be used as identifiers. Examples of keywords in C++ include int, char, float, for, while, if, else, and return.
- **Identifiers:** Identifiers are names used to identify variables, functions, classes, and other elements in a C++ program. Identifiers must start with a letter or an underscore, and they can be followed by letters, digits, or underscores.
- **Constants:** Constants are values that are assigned to variables or used in expressions. Constants in C++ can be integer constants, floating-point constants, character constants, or string literals.
- **Operators:** Operators are symbols that perform operations on variables and values in a program. C++ supports several types of operators, including arithmetic operators, relational operators, logical operators, and bitwise operators.
- **Punctuation:** Punctuation marks are symbols that are used to separate elements in a program, such as semicolons, commas, parentheses, brackets, and braces. Punctuation marks are used to specify the syntax of a program and to delimit blocks of code.

# C++ Tokens…

- A C++ program includes various tokens.

```cpp
// Includes the standard I/O library
#include <iostream>
using namespace std; // Uses the std namespace

// Defines a simple function with a parameter
  int add(int x) {
    return x + 1;
  }

int main() {
  // Declares a constant integer variable
  const int number = 100;

  // Prints a message with a newline character
  cout << "Hello, C++ Tokens!\n";

 // Prints a given number with a newline
  cout << "Given Number: " << number << endl;

```

# C++ Tokens

```
Hello, C++ Tokens!
Given Number: 100
Increament of 100 gives: 101
The condition is true.
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

```
20    // Calls the function and prints the result
21     cout << "Increament of " << number << " gives: " << add
           (number) << endl;
22
23    // Defines a boolean variable and checks its value
24     bool isTrue = true;
25     if (isTrue) {
26       cout << "The condition is true." << endl;
27     }
28
29     // Declares a floating-point variable and assigns a value
30     double pi = 3.14159;
31
32     // Uses a loop to iterate 5 times
33     for (int i = 0; i < 5; ++i) {
34       cout << "Iteration " << i + 1 << endl;
35     }
36
37     // Returns 0 to indicate successful program execution
38     return 0;
39   }
```

# Simple C++ Programs…

- Addition of Two Numbers:

```cpp
#include <iostream>
int main() {
    int num1, num2, sum;

    std::cout << "Enter first number: ";
    std::cin >> num1;

    std::cout << "Enter second number: ";
    std::cin >> num2;

    sum = num1 + num2;

    std::cout << "Sum: " << sum << std::endl;

    return 0;
}
```

```
Enter first number: 12
Enter second number: 23
Sum: 35
```

# Simple C++ Programs…

- Simple Calculator:

```cpp
#include <iostream>
using namespace std;
int main() {
    char op;
    double num1, num2;

    cout << "Enter operator (+, -, *, /): ";
    cin >> op;

    cout << "Enter first number: ";
    cin >> num1;

    cout << "Enter second number: ";
    cin >> num2;

    switch(op) {
        case '+':
            cout << "Result: " << num1 + num2 << endl;
            break;
```

# Simple C++ Programs…

- Simple Calculator:

```
Enter operator (+, -, *, /): *
Enter first number: 12
Enter second number: 5
Result: 60
```

```
20          case '-':
21              cout << "Result: " << num1 - num2 << endl;
22              break;
23          case '*':
24              cout << "Result: " << num1 * num2 << endl;
25              break;
26          case '/':
27              if(num2 != 0) {
28                  cout << "Result: " << num1 / num2 << endl;
29              } else {
30                  cout << "Error: Division by zero" << endl;
31              }
32              break;
33          default:
34              cout << "Error: Invalid operator" << endl;
35      }
36      return 0;
37  }
```

# Simple C++ Programs…

- Factorial Calculation:

```cpp
#include <iostream>
int main() {
    int num;
    long long factorial = 1;

    std::cout << "Enter a positive integer: ";
    std::cin >> num;

    if(num < 0) {
        std::cout << "Error: Factorial is not defined for negative
            numbers." << std::endl;
    } else {
        for(int i = 1; i <= num; ++i) {
            factorial *= i;
        }
        std::cout << "Factorial of " << num << " is " << factorial
            << std::endl;
    }

    return 0;
}
```

```
Enter a positive integer: 7
Factorial of 7 is 5040
```

# Simple C++ Programs

- Fibonacci Series:

```cpp
#include <iostream>
int main() {
    int n;
    std::cout << "Enter the number of terms: ";
    std::cin >> n;
    int first = 0, second = 1, next;
    std::cout << "Fibonacci Series: ";
    for(int i = 0; i < n; ++i) {
        if(i == 0) {
            std::cout << first << " ";
            continue;
        }
        if(i == 1) {
            std::cout << second << " ";
            continue;
        }
        next = first + second;
        first = second;
        second = next;

        std::cout << next << " ";
    }
    std::cout << std::endl;
    return 0;
}
```

```
Enter the number of terms: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

Lecture 2

C++ Basics

# THE END