Functions in C++

By:
freeCSIT

C++ course with PDF Notes

Functions in C++

Topics:-

- Definition
- Types
- Miscellaneous
- Benefits of Using Functions
- Components
- A Simple Function
- Syntax

What is Function in C++ ?

Definition & types

A function is defined as a group of statements that performs a task together and the user can execute them whenever required. There are two types of functions: predefined functions and user defined functions.

Predefined functions

User defined functions

Key Components in C++

Components

- return_type
- function_name
- parameter_list
- body

```
returnType  functionName(parameters) {
   // Function body
   return value; // Optional for non-void functions
}
```

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

// Function declaration
void greeting() {
    cout << "Hello, World!" << endl;
}

int main() {
    greeting(); // Function call
    return 0;
}
```

```
Hello, World!
```

Key Components in C++

Components

return_type

function_name

parameter_list

body

```
returnType  functionName(parameters) {
   // Function body
   return value; // Optional for non-void functions
}
```

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

// Function to add two numbers
int add(int a, int b) {
    return a + b;
}

int main() {
    int x = 5, y = 10;
    cout << "Sum: " << add(x, y) << endl; // Function call
    return 0;
}
```

```
Sum: 15
```

Syntax of function in C++

```
returnType  functionName (parameters) {

    // Function body

    return value; // Optional for non-void functions

}
```

Function Parameters in C++

Parameters

Arguments

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

void myFunction(string courseName) {
  cout << courseName << " with freeCSIT " <<endl;
}

int main() {
  myFunction("C");
  myFunction("C++");
  myFunction("DSA");
  return 0;
}
```

Output:

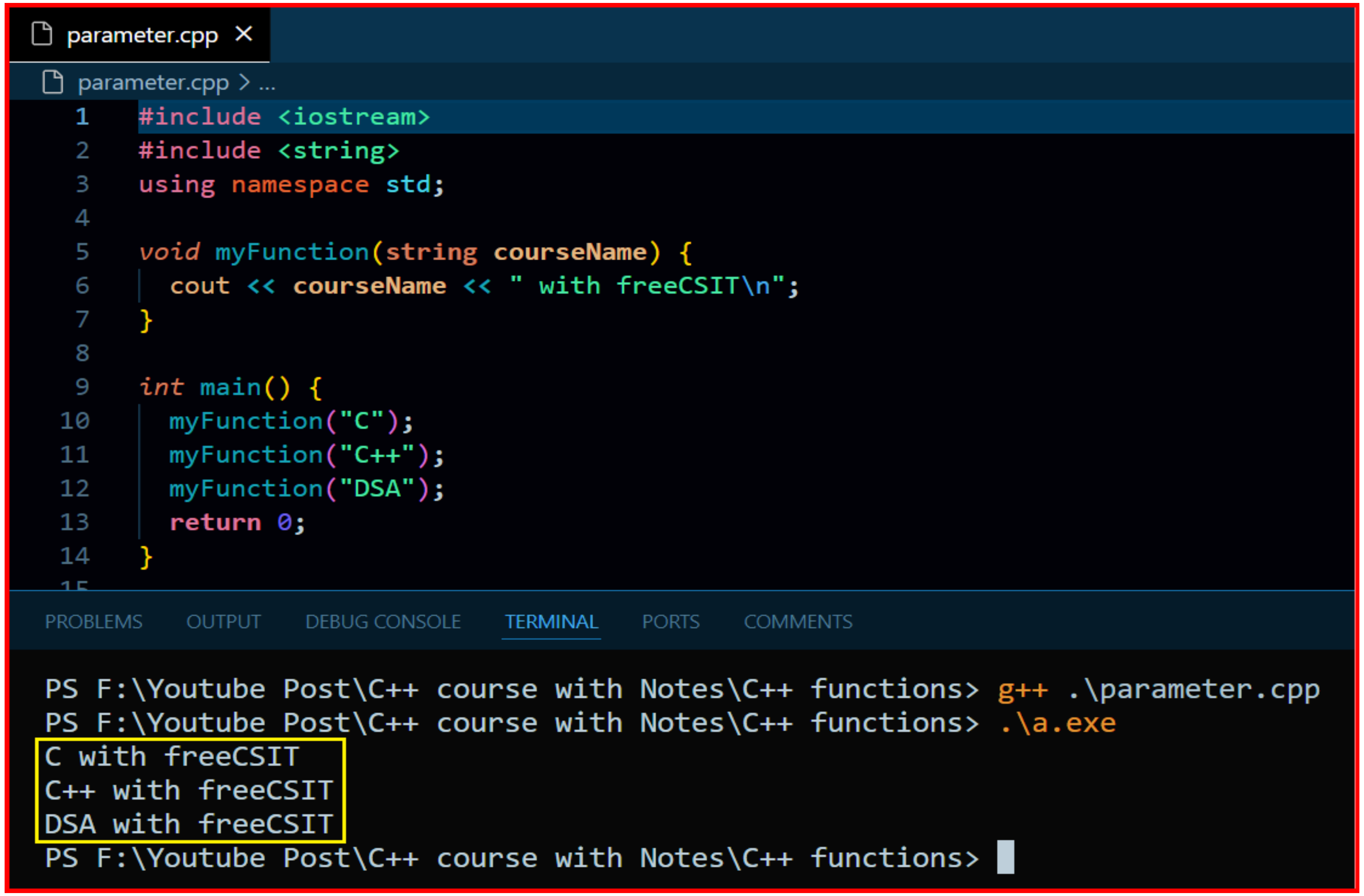
```
C with freeCSIT
C++ with freeCSIT
DSA with freeCSIT
```

freeCSIT

Code Demo in VS code

Function Parameters in C++

C++ course with Notes


parameter.cpp

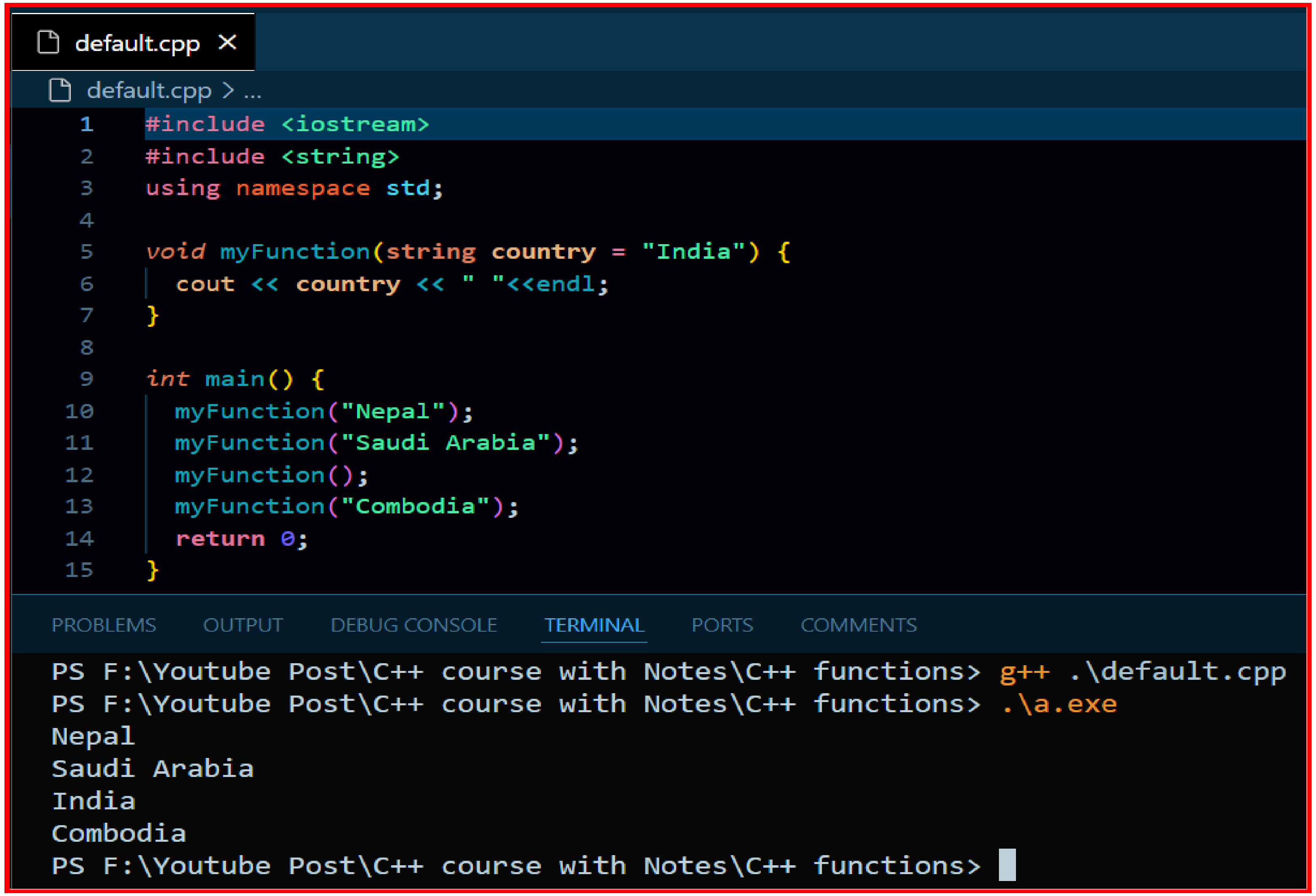
parameter.cpp > ...

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

void myFunction(string courseName) {
  cout << courseName << " with freeCSIT\n";
}

int main() {
  myFunction("C");
  myFunction("C++");
  myFunction("DSA");
  return 0;
}
```

```
PS F:\Youtube Post\C++ course with Notes\C++ functions> g++ .\parameter.cpp
PS F:\Youtube Post\C++ course with Notes\C++ functions> .\a.exe
C with freeCSIT
C++ with freeCSIT
DSA with freeCSIT
PS F:\Youtube Post\C++ course with Notes\C++ functions>
```

Code Demo in VS code

Default Parameters in C++



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

void myFunction(string country = "India") {
  cout << country << " "<<endl;
}

int main() {
  myFunction("Nepal");
  myFunction("Saudi Arabia");
  myFunction();
  myFunction("Combodia");
  return 0;
}
```

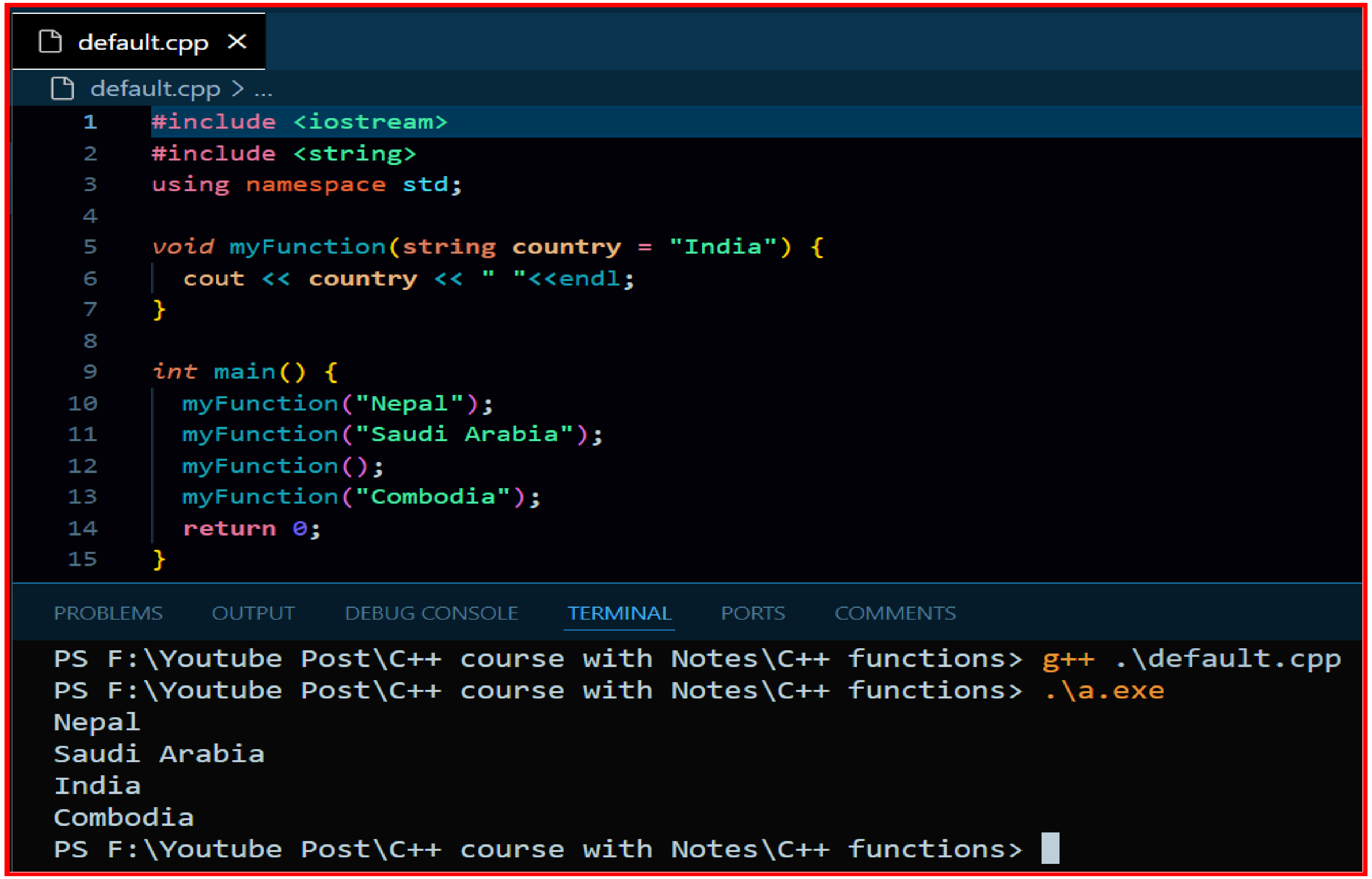
```
PS F:\Youtube Post\C++ course with Notes\C++ functions> g++ .\default.cpp
PS F:\Youtube Post\C++ course with Notes\C++ functions> .\a.exe
Nepal
Saudi Arabia
India
Combodia
PS F:\Youtube Post\C++ course with Notes\C++ functions>
```

freeCSIT

Code Demo in VS code

Multiple Parameters in C++

C++ course with Notes



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

void myFunction(string country = "India") {
  cout << country << " "<<endl;
}

int main() {
  myFunction("Nepal");
  myFunction("Saudi Arabia");
  myFunction();
  myFunction("Combodia");
  return 0;
}
```

```
PS F:\Youtube Post\C++ course with Notes\C++ functions> g++ .\default.cpp
PS F:\Youtube Post\C++ course with Notes\C++ functions> .\a.exe
Nepal
Saudi Arabia
India
Combodia
PS F:\Youtube Post\C++ course with Notes\C++ functions>
```

A Simple Function in C++

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

// Function declaration
void greeting() {
    cout << "Hello, World!" << endl;
}

int main() {
    greeting(); // Function call
    return 0;
}
```

Function with Parameters and return value

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

// Function to add two numbers
int add(int a, int b) {
   return a + b;
}

int main() {
   int x = 5, y = 10;
   cout << "Sum: " << add(x, y) << endl; // Function call
   return 0;
}
```

Advantages of Function in C++

Code Reusability: Write once, use multiple times.
Modularity: Breaks the program into manageable parts.
Ease of Debugging: Isolates and tests smaller pieces of code.
Improves Readability: Better organized and structured code.

Passing Data to Functions

Pass by Value

One method of passing data to function is passing by value. In this way of passing variables, a copy of the variable (main program) is created during function call with the name specified in the function and initialized with the value in the original variable. All the operations in the function are then performed on the function variable. The values in the variables declared in the main program remain unchanged by the function operations.

Pass by Reference

Another alternative to passing arguments is passing by reference. When we pass argument by reference, no copy of the variable is created. However, the variables in the main program are referred to by different name in the function. Since no copy is created, when the values in the function variables are modified, the values in the variables in the main program are also modified. Pass by reference provides an easy mechanism for modifying the variables by functions and also enables to return multiple variables.
To pass arguments by reference, all the variable names in the argument list should be prefixed with & (ampersand) or address of operator when function is declared and defined.

Passing Data to Functions

Pass by Value

```
#include <iostream>
void display(int num) {
  num += 10; // Changes won't affect the original variable
  std::cout << "Value inside function: " << num << std::endl;
}

int main() {
  int x = 5;
  display(x);
  std::cout << "Value in main: " << x << std::endl; // x remains unchanged
  return 0;
}
```

Passing Data to Functions

Pass by Reference

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

// Function to swap two numbers
void swap(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x = 3, y = 7;
    cout << "Before Swap: x = " << x << ", y = " << y << endl;
    swap(x, y); // Function call
    cout << "After Swap: x = " << x << ", y = " << y << endl;
    return 0;
}
```

Passing Data to Functions

Pass by Reference

```
#include <iostream>
void increment(int &num) {
    num += 10; // Affects the original variable
}

int main() {
    int x = 5;
    increment(x);
    std::cout << "Value after increment: " << x << std::endl; // x is updated
    return 0;
}
```

Function Overloading in C++

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

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

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

int main() {
   cout << multiply(2, 3) << endl;        // Calls int version
   cout << multiply(2.5, 3.5) << endl;    // Calls double version
   return 0;
}
```

Function Scope in C++

Local Scope

A variable created inside a function belongs to the *local scope* of that function, and can only be used inside that function:

A **local variable** cannot be used outside the function it belongs to.
If you try to access it outside the function, an error occurs:

Global Scope

A variable created outside of a function, is called a **global variable** and belongs to the *global scope*.
Global variables are available from within any scope, global and local:

Recursion in C++

Recursion is a powerful concept in C++ (and other programming languages) where a function calls itself to solve smaller instances of a problem until it reaches a base case. It's particularly useful for problems that can be broken down into smaller, similar subproblems, such as calculating factorials, generating Fibonacci numbers, or traversing data structures like trees.

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

// Recursive function to calculate factorial
int factorial(int n) {
   if (n == 0) return 1; // Base case
   return n * factorial(n - 1); // Recursive case
}

int main() {
   int num = 5;
   cout << "Factorial of " << num << " is " << factorial(num) << endl;
   return 0;
}
```

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

// Recursive function to calculate nth Fibonacci number
int fibonacci(int n) {
   if (n <= 1) return n; // Base case
   return fibonacci(n - 1) + fibonacci(n - 2); // Recursive case
}

int main() {
   int n = 6; // Find the 6th Fibonacci number
   cout << "Fibonacci number at position " << n << " is " << fibonacci(n) << endl;
   return 0;
}
```

Thanks for watching

Please Subscribe