Programming Fundamentals with C++

Lecture 18 – Functions

Dr. Muhammad Sajjad

RA: Asad Ullah

Overview

- Local and Global Functions
- Reference Parameters
- Default Arguments
- Command Line Arguments
- Inline Functions
- Function Overloading
- Function Templates

Local and Global Functions

- **Local Functions:**
  - Functions declared inside another function are not allowed in C++.
  - But functions declared inside main() before use are considered local to main().
- **Global Functions:**
  - Declared outside main() and accessible from anywhere in the program.

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

// Global function
void greet() {
   cout << "Hello from a global function!" << endl;
}

int main() {
   greet();  // Calling global function
   greeting(); // Calling local function

// Local function
void greeting() {
   cout << "Hello from a local function!" << endl;
}
   return 0;
}
```

Use Case: Used for functions that need to be accessed from multiple parts of a program.

Reference Parameters

- Instead of passing a copy of a variable, we pass a reference using &.
- This allows the function to modify the original variable.
- **Example:**

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

void updateValue(int &num) {  // Pass by reference
    num = num * 2;
}

int main() {
    int x = 10;
    updateValue(x);
    cout << "Updated value: " << x << endl;  // Output: 20
    return 0;
}
```

Use Case: Used for modifying values without returning them.

Default Arguments

- If a function argument is **not provided**, it uses the **default value**.
- **Example:**

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

void greet(string name = "Guest") {  // Default argument
    cout << "Hello, " << name << "!" << endl;
}

int main() {
    greet();         // Output: Hello, Guest!
    greet("Ali");    // Output: Hello, Ali!
    return 0;
}
```

Use Case: Used for **optional parameters** in functions.

Command-Line Arguments

- Used to take input when running a program from the command line.
- Uses int argc (argument count) and char* argv[] (argument values).
- **Example:**

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

int main(int argc, char* argv[]) {
    cout << "Number of arguments: " << argc << endl;
    for(int i = 0; i < argc; i++) {
        cout << "Argument " << i << ": " << argv[i] << endl;
    }
    return 0;
}
```

Use Case: Used in **CLI applications** to accept user input.

Inline Functions

- inline keyword replaces function calls with actual function code.
- Makes execution faster by avoiding function calls.
- **Example:**

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

inline int square(int x) { return x * x; }

int main() {
    cout << "Square of 5: " << square(5) << endl;
    return 0;
}
```

Use Case: Used for **small, frequently used functions**.

Function Overloading

- Multiple functions **with the same name but different parameters**.
- **Example:**

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

void print(int x) {
    cout << "Integer: " << x << endl;
}
void print(double x) {
    cout << "Double: " << x << endl;
}
void print(string x) {
    cout << "String: " << x << endl;
}

int main() {
    print(10);
    print(10.5);
    print("Hello");
    return 0;
}
```

Use Case: Used for **multiple versions of a function**.

Function Templates

- Allows functions to **work with any data type**.
- **Example:**

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

template <typename T>
T add(T a, T b) {
    return a + b;
}

int main() {
    cout << "Sum (int): " << add(5, 10) << endl;
    cout << "Sum (double): " << add(2.5, 3.7) << endl;
    return 0;
}
```

Use Case: Used in generic programming (e.g., vector<int>, vector<double> in STL).

Summary Table

Feature	Purpose	Example Usage
Local & Global Variable	Scope of functions	Global utility functions
Reference Parameter	Modify original variable	Swapping values
Default Arguments	Provide default values	Optional parameters
Command-line Arguments	Input from terminal	CLI applications
Inline Functions	Reduce function call overhead	Fast small functions
Function Overloading	Same function name, different parameters	Multiple versions of a function
Function Templates	Generic programming	Works for different data types

Thank You