

Chapter 5: Functions in C++

Mastering modular programming, code reusability, parameters, and return types for robust software engineering.



Learning Objectives



Define Functions

Understand modular programming concepts and syntax.



Declaration & Calling

Build prototypes, pass parameters, and invoke functions.



Scope & Storage

Analyze local vs global variables and execution stacks.

CORE CONCEPT

What is a Function?

A function is a self-contained block of code designed to perform a specific, well-defined task. Instead of writing the same code repeatedly, programmers encapsulate logic inside functions.

- ✓ **Code Reusability:** Write once, invoke anytime.
- ✓ **Easy Maintenance:** Isolate bugs quickly within modules.
- ✓ **Improved Readability:** Clean, structured program logic.

```
// Function Anatomy
return_type function_name(parameters) {
    // Body of the function
    return value;
}
```

CLASSIFICATION

Types of Functions in C++



Built-in (Library) Functions

Pre-written functions provided by C++ standard libraries.

```
sqrt(), pow(), abs(), strlen()
```



User-Defined Functions

Custom routines crafted by programmers to solve specific tasks.

```
calculateSum(), displayMenu()
```

Declaration & Definition

A function must be declared before it is used. The **prototype** informs the compiler about the function name, return type, and parameters.

- **Prototype:** Declared before `main()` .
- **Definition:** Contains actual executable code block.

```
// Complete Structure Example
int add(int a, int b); // Prototype
int main() {
    int res = add(5, 10);
}
int add(int a, int b) { // Definition
    return (a + b);
}
```

Arguments vs Parameters



Parameters (Formal)

Variables defined in the function header that receive data values during function invocation.



Arguments (Actual)

Actual values or variables passed into the function call from the calling environment (e.g., `main()`).

Pass by Value vs Pass by Reference

Pass by Value

Copies actual value into formal parameter. Changes inside function do not affect original variable.

```
void modify(int x);
```

Pass by Reference

Passes memory address using reference operator (&). Modifications directly alter original variable.

```
void modify(int &x);
```

Local vs Global Variables

Local Variables

Declared inside a function or block. Accessible only within that specific function scope. Destroyed upon exit.

Global Variables

Declared outside all functions. Accessible by any function throughout the entire program execution lifecycle.

IMPLEMENTATION

Function Implementation in C++

```
#include
using namespace std;

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

int main() {
    int result = multiply(6, 7);
    cout << "Product: " << result;
    return 0;
}
```

REVIEW

Chapter Summary

- ✓ Functions break complex software development into manageable, modular components.
- ✓ Proper prototyping and return typing ensure strict compiler validation.
- ✓ Understanding scope and parameter passing safeguards program state and memory integrity.

KARACHI EDU

Questions & Discussion

Prepare your queries for our coaching center lab session!
