Lecture 6: Functions in C++

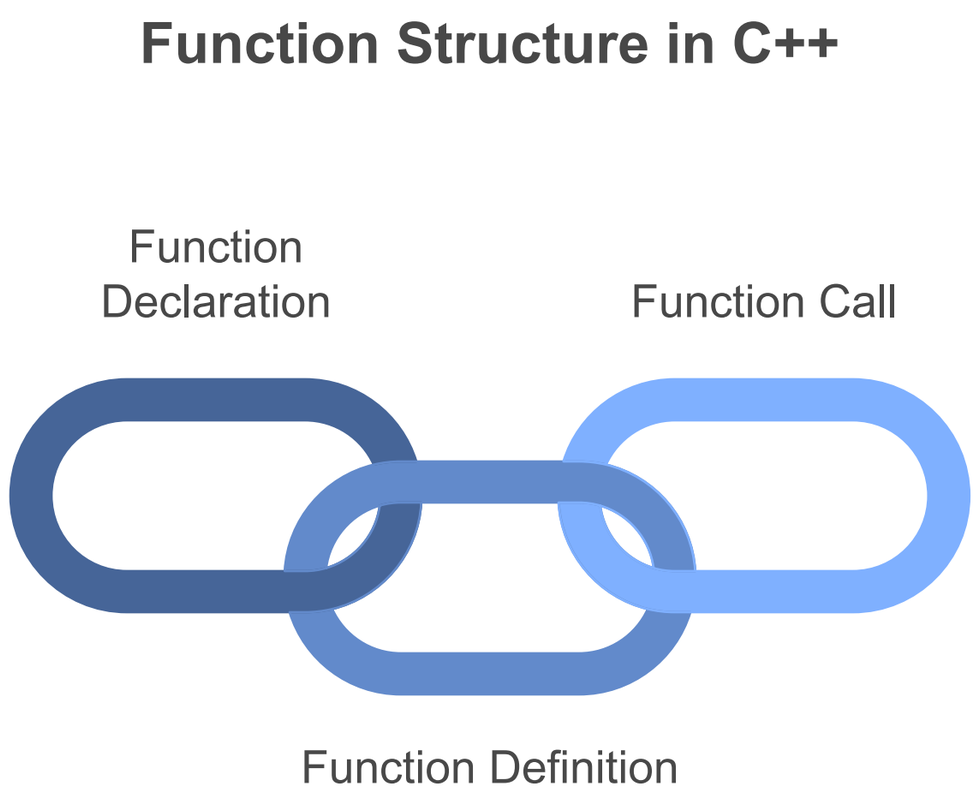
Overview
Functions are fundamental building blocks in programming, allowing us to break down complex problems into smaller, manageable pieces of code. This lecture covers C++ functions, from built-in utility functions to custom function creation, parameters, return types, and variable scopes.
1. What is a Function?

A **function** is a block of code designed to perform a specific task. Functions help modularize code, making it more readable, reusable, and organized. In C++, functions come in two primary forms:

- **Built-in functions:** Functions provided by the C++ Standard Library.
- **User-defined functions:** Custom functions created to solve specific problems.

Example Structure:
A function typically includes:

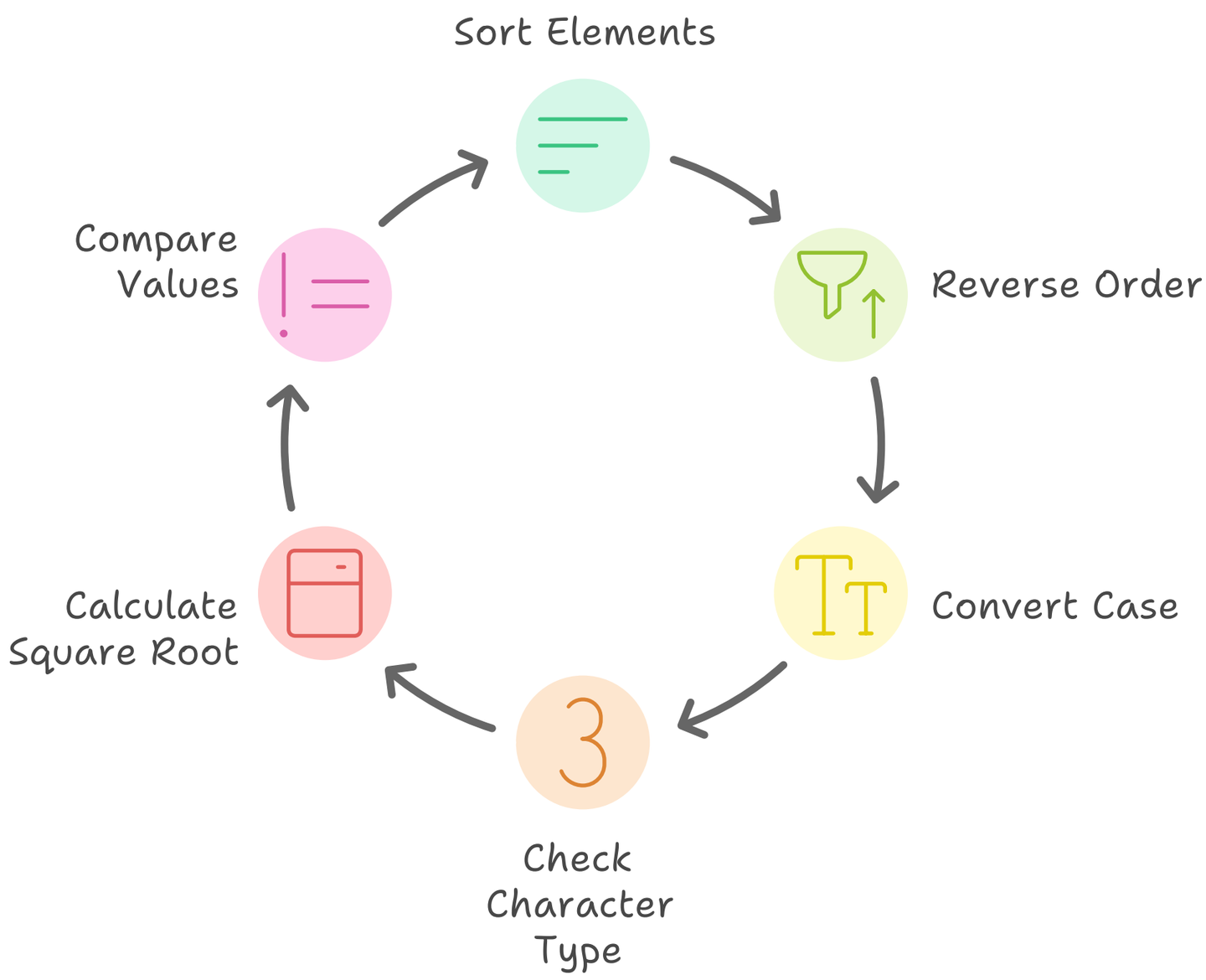
1. **Function Declaration** (Prototype)
2. **Function Definition** (Body)
3. **Function Call**



2. Built-in Functions

C++ provides a set of built-in functions for common operations. Let's go through some of them:

1. **sort()** - Sorts elements in a range (e.g., an array or vector).
2. **reverse()** - Reverses the order of elements in a range.
3. **tolower()** - Converts a character to lowercase.
4. **toupper()** - Converts a character to uppercase.
5. **isalpha()** - Checks if a character is alphabetic.
6. **isdigit()** - Checks if a character is a digit.
7. **isalnum()** - Checks if a character is alphanumeric.
8. **ispunct()** - Checks if a character is punctuation.
9. **sqrt()** - Calculates the square root of a number.
10. **max() / min()** - Finds the maximum or minimum of two values.



Example:

```
#include <iostream>
#include <algorithm>
#include <cctype>
#include <cmath>
using namespace std;

int main() {
    char ch = 'A';
    cout << "tolower('A'): " << tolower(ch) << endl;
    cout << "toupper('a'): " << toupper('a') << endl;
    cout << "isalpha('A'): " << isalpha(ch) << endl;
    cout << "isdigit('1'): " << isdigit('1') << endl;
    cout << "sqrt(16): " << sqrt(16) << endl;
    cout << "max(3, 7): " << max(3, 7) << endl;
    cout << "min(3, 7): " << min(3, 7) << endl;

    return 0;
}
```

3. Function Calling

A function must be called to execute its code. Calling a function transfers control to the function, and after it completes, control returns to the calling point.

```
void greet() {
    cout << "Hello, welcome to C++ functions!" << endl;
}

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

```
}
```

4. Function Declaration and Definition

The **function declaration** tells the compiler about a function's name, return type, and parameters without implementing its body. The **function definition** provides the actual code for the function.

```
// Function Declaration (Prototype)
int add(int a, int b);

// Function Definition
int add(int a, int b) {
    return a + b;
}

int main() {
    cout << "Sum: " << add(5, 3) << endl;
    return 0;
}
```

5. Function Parameters

Parameters allow functions to accept data from the calling function. In C++, function parameters can be of any data type, such as **int**, **double**, or **char**.

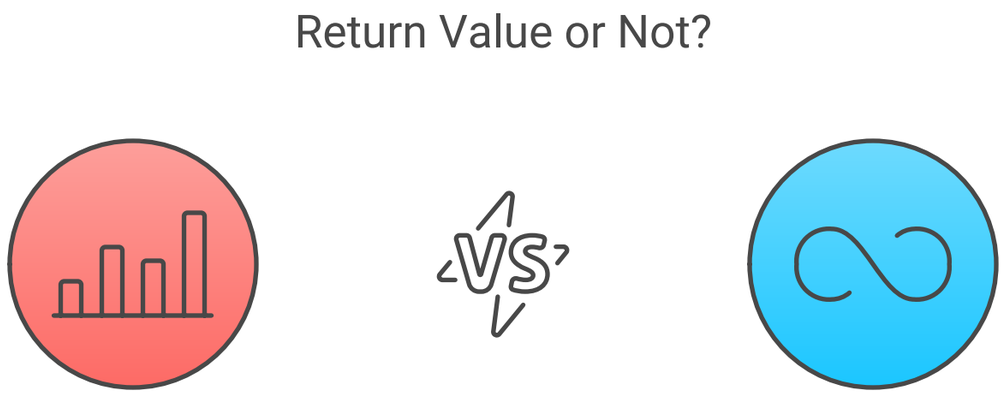
```
void printSum(int a, int b) {
    cout << "Sum: " << a + b << endl;
}

int main() {
    printSum(4, 5); // Passing 4 and 5 as arguments
    return 0;
}
```

6. Types of Functions

There are two primary types of functions in C++:

- **Returning Data:** Functions that return a value using the **return** keyword.
- **Void Functions:** Functions that do not return any data; they have a **void** return type.



Examples:

1. **Function Returning Data**

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

int main() {
    int result = multiply(5, 3);
    cout << "Product: " << result << endl;
```

```
    return 0;
}
```

2. **Void Function**

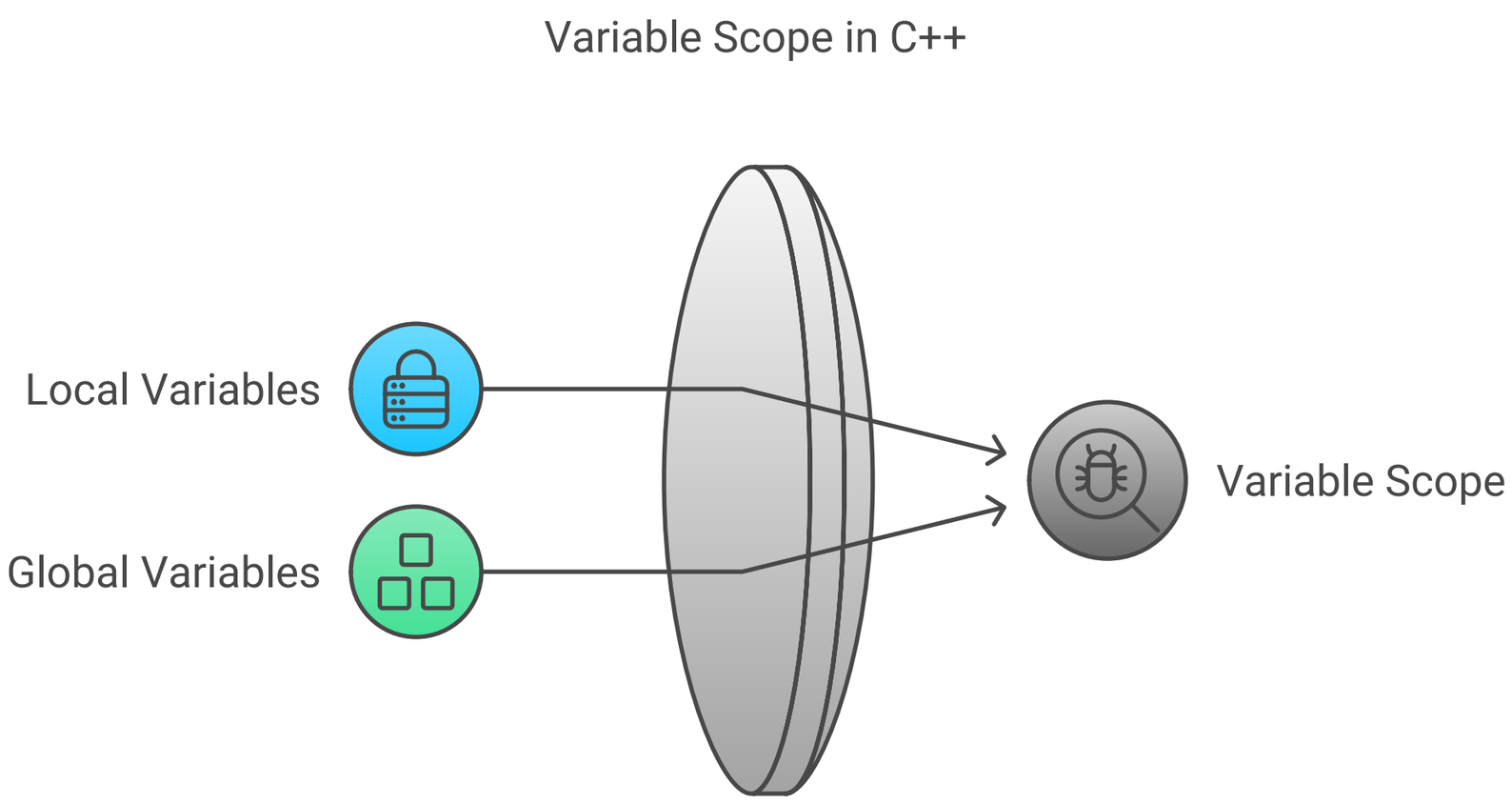
```
void greet() {
    cout << "Hello!" << endl;
}

int main() {
    greet();
    return 0;
}
```

7. Local and Global Variables

Variables can be classified based on their scope:

- **Local Variables**: Declared inside a function or block, accessible only within that function or block.
- **Global Variables**: Declared outside all functions, accessible from any function within the same file.



Example:

```
int globalVar = 10; // Global variable

void display() {
    int localVar = 5; // Local variable
    cout << "Global: " << globalVar << ", Local: " << localVar << endl;
}

int main() {
    display();
    // cout << localVar; // Error: localVar is not accessible here
    return 0;
}
```

Summary

In C++, functions help break down code into reusable blocks, enabling cleaner and more maintainable programs. By understanding built-in functions, function types, parameter passing, and variable scope, you can better organize and structure your code for efficiency and readability.