Al-Mustaqbal University
College of Sciences
Department of Cybersecurity

كليـــة العلـــوم
قسم الأمن السيبراني

Subject: Object Oriented Programming (OOP)

Second Stage

Lecturer: Dr. Abdulkadhem A. Abdulkadhem

Lecture (2)

Functions and Parameter Transmission

2024-2025

1. Introduction to Functions

What is a Function?

A function is a block of code designed to perform a specific task, reusable throughout a program. C++ allows flexible function definitions and parameter handling.

Parameter Transmission in Functions:

In C++, parameters can be passed in multiple ways:

- **Pass by Value**: A copy of the argument is passed.

- **Pass by Reference**: The actual argument is passed, allowing the function to modify the argument.

2. Function Overloading

Definition: Function overloading allows multiple functions to have the same name with different signatures (i.e., different parameter types or numbers of parameters).

Benefits: Provides better code readability and reusability.

Example Code:

```
#include <iostream>
using namespace std;
// Function to calculate the area of a rectangle
int area(int length, int width) {
   return length * width;
}

// Overloaded function to calculate the area of a circle
double area(double radius) {
   return 3.1415 * radius * radius;
}
```

```
int main() {
    int length = 5, width = 10;
    double radius = 7.5;

    cout << "Area of rectangle: " << area(length, width) << endl;
    cout << "Area of circle: " << area(radius) << endl;

    return 0;
}
```

Explanation:
The 'area' function is overloaded: one version takes two 'int' parameters, and the other takes a 'double' parameter. The correct function is selected based on the argument type.

3. Inline Functions

Definition: Inline functions are expanded in line where they are called, which can reduce function call overhead, especially for small functions.

When to Use: Inline functions should be used for small, frequently called functions.

Example Code:

```
#include <iostream>
using namespace std;
// Inline function to add two numbers
inline int add(int a, int b) {
    return a + b;
}

int main() {
    int x = 10, y = 20;
    cout << "Sum: " << add(x, y) << endl;
```

```
    return 0;
}
```

Explanation:
The 'add' function is declared as 'inline'. When called, the compiler replaces the function call with the function's code, which can improve performance for small functions.

4. Default Arguments

Definition: Default arguments allow function parameters to have default values if no arguments are passed during the function call.

Advantages: Simplifies function calls and improves code readability.

Example Code:

```
#include <iostream>
using namespace std;
// Function to print a message with a default argument
void greet(string name = "Guest") {
    cout << "Hello, " << name << "!" << endl;
}

int main() {
    greet();          // Uses default argument
    greet("Alice");    // Passes 'Alice' as argument

    return 0;
}
```

Explanation:
The 'greet' function has a default argument 'Guest'. If no argument is passed, the default value is used. Otherwise, the provided argument overrides the default value.

5. Pass by Reference

Definition: Passing by reference allows a function to modify the original argument. This is achieved by passing a reference to the argument rather than a copy.

Benefits: Saves memory (no copy is made) and allows modification of the caller's variables.

Example Code:

```
#include <iostream>
using namespace std;
// Function to swap two integers using pass by reference
void swap(int &a, int &b) {
   int temp = a;
   a = b;
   b = temp;
}

int main() {
   int x = 5, y = 10;
   cout << "Before swap: x = " << x << ", y = " << y << endl;

   swap(x, y);  // Passing variables by reference
   cout << "After swap: x = " << x << ", y = " << y << endl;

   return 0;
}
```

Explanation:
The 'swap' function takes two reference parameters ('&a' and '&b'). Changes made to 'a' and 'b' within the function affect the original 'x' and 'y' variables in 'main'.

6. Return by Reference

Definition: Returning by reference allows a function to return a reference to a variable rather than a copy, enabling the caller to modify the returned variable directly.

When to Use: Used when you want the function to return a variable that can be modified by the caller.

Example Code:

```
#include <iostream>
using namespace std;
// Function that returns a reference to a variable
int& getLargest(int &a, int &b) {
   return (a > b) ? a : b;
}

int main() {
   int x = 5, y = 10;
   cout << "Before modification: x = " << x << ", y = " << y << endl;

   getLargest(x, y) = 100;  // Modifying the largest number by reference
   cout << "After modification: x = " << x << ", y = " << y << endl;

   return 0;
}
```

Explanation:
The 'getLargest' function returns a reference to the largest of two integers. In 'main', the returned reference is used to directly modify the largest variable.

Summary

1. Function Overloading allows multiple functions with the same name but different parameters.

2. Inline Functions can reduce the overhead of small function calls by expanding them inline.
3. Default Arguments provide default values for function parameters.
4. Pass by Reference allows a function to modify the caller's arguments directly, saving memory and time.
5. Return by Reference enables a function to return a reference to a variable, allowing modifications outside the function.

Homework Assignment

Instructions:

- Complete each task as specified.
- Write clean, well-commented C++ code for each task.
- Ensure that your code compiles and runs correctly.
- Due Date: 1 week.
- Submit your homework to the google form [https://forms.gle/aLc1JinRpfKmtwg69].

Task 1: Function Overloading

Write an overloaded function named *calculate* to perform the following operations:

- For two integer inputs: Return the sum of the two numbers.
- For two floating-point inputs: Return the product of the two numbers.
- For three integer inputs: Return the largest of the three numbers.

```
cout << calculate(5, 10);      // Output: 15 (integer sum)
cout << calculate(2.5, 4.0);   // Output: 10.0 (floating-point product)
cout << calculate(3, 7, 2);    // Output: 7 (largest of three integers)
```

Task 2: Default Arguments

Write a function named *printMessage* that takes two parameters: message (a string) and times (an integer). The function should print the message the specified number of times. If no times argument is provided, the default value should be 3.

Example:

```
printMessage("Hello!");       // Prints "Hello!" 3 times
printMessage("Hi!", 5);       // Prints "Hi!" 5 times
```