Lecture 1

Object Oriented Programming

Titles

1. Review of Functions and Parameter Transmission.
2. Function Overloading and Inline Functions.
3. Default Arguments, Pass by Reference, Return by Reference.

Learning Objectives

By the end of this lecture, students will be able to:

1- Define the concept of functions in **C++**.

2- Distinguish between different parameter passing techniques **(by value, by reference)**.

3- Apply default arguments in function definitions.

4- Understand the concept of function overloading.

5- Implement inline functions to optimize execution.

Theoretical Part

1. Review of Functions

Definition: A function is a block of code that performs a specific task.

General Syntax:

```
return_type function_name(parameters) {
    // body of function
}
```

Theoretical Part

2. Parameter Transmission

- **Pass by Value:** A copy of the variable is passed; changes inside the function do not affect the original variable.
- **Pass by Reference :** The address of the variable is passed; changes inside the function directly modify the original variable.
- **Return by Reference:** A function can return a reference instead of a value, allowing the returned value to be used as an alias of the original variable.

Theoretical Part

3. Default Arguments

- A parameter may be assigned a default value when the function is defined.
- If no argument is passed during a function call, the default value will be used.

Example:

```
int sum(int a, int b = 10) {
    return a + b;
}

cout << sum(5);   // Output: 15
cout << sum(5,7); // Output: 12
```

Theoretical Part

4. Function Overloading

Multiple functions can have the same name but must differ in the number or type of parameters.

Example:

```
int add(int a, int b) {
    return a + b;
}

double add(double a, double b) {
    return a + b;
}
```

Theoretical Part

5. Inline Functions

- Inline functions instruct the compiler to replace the function call with the actual function code during compilation.
- This reduces the overhead of function calls, but is recommended only for small functions.

Example:

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

Practical Illustrations

Example 1 – Pass by Value vs Pass by Reference

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

void byValue(int x) {
    x = x + 10;
}

void byReference(int &y) {
    y = y + 10;
}

int main() {
    int a = 5, b = 5;
    byValue(a);      // No change
    byReference(b);  // Changes original value
    cout << "a = " << a << endl; // 5
    cout << "b = " << b << endl; // 15
}
```

Practical Illustrations

Example 2 – Function Overloading

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

int multiply(int x, int y) {
    return x * y;
}

double multiply(double x, double y) {
    return x * y;
}

int main() {
    cout << multiply(3, 4) << endl;      // 12
    cout << multiply(2.5, 4.0) << endl;  // 10.0
}
```

Practical Illustrations

Example 3 – Inline Function

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

inline int cube(int n) {
    return n * n * n;
}

int main() {
    cout << cube(3) << endl; // 27
}
```

Assignments

1. Write an inline function to compute the absolute value of an integer.
2. Write two overloaded functions with the same name: one that calculates the area of a rectangle (length × width) and another that calculates the area of a square (side × side).
3. Implement a program that demonstrates the difference between Pass by Value and Pass by Reference by doubling the value of a variable.

Thank You...