Object Oriented Programming Lab 01

Course: Object Oriented Programming (CL1004) **Semester**: Spring 2025
Instructor: Muhammad Monis

Note:

- Maintain discipline during the lab.
- Listen and follow the instructions as they are given.
- Just raise hand if you have any problem.
- Completing all tasks of each lab is compulsory.
- Get your lab checked at the end of the session.

Introduction to C++

Skeleton of C++ Program

A C++ program is structured in a specific and particular manner. In C++, a program is divided into the following three sections:

1. Standard Libraries Section
2. Main Function Section
3. Function Body Section

For example, let's look at the implementation of the Hello World program:

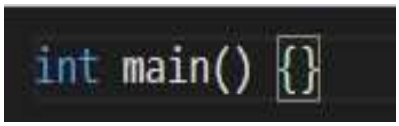
```cpp
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!" << endl;
  return 0;
}
```

Standard Libraries Section

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

- #include is a specific preprocessor command that effectively copies and pastes the entire text of the file, specified between the angle brackets, into the source code.
- The file <iostream>, which is a standard file that should come with the C++ compiler, is short for input-output streams. This command contains code for displaying and getting an input from the user.
- namespace is a prefix that is applied to all the names in a certain set. iostream file defines two names used in this program - cout and endl.

Main Function Section



- The starting point of all C++ programs is the main function.
- This function is called by the operating system when your program is executed by the computer.
- {Signifies the start of a block of code, and} signifies the end.

Input/Output in C++ Program

- C++ is very similar to the C Language.
- For the input/output stream we use <iostream> library (in C it was <stdio>).
- For taking input and out we cout and cin (in C it was printf and scanf).
- cout uses insertion (<<) operator.
- cin uses extraction (>>) operator.

Sample C++ Code

```
#include <iostream>

using namespace std;

int main()
{
int var = 0;
cout << "Enter an Integer value: ";
cin >>var;
}
```

Variables & Data Types in C++

While writing a program in any language, you need to use various variables to store various information. Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

You may like to store information of various data types like character, wide character, integer, floating point, double floating point, Boolean etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

Primitive Built-in Types

C++ offers the programmer a rich assortment of built-in as well as user defined data types.

Following table lists down seven basic C++ data types:

Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void
Wide character	wchar_t

Several of the basic types can be modified using one or more of these type modifiers:

- signed
- unsigned
- short
- long

The following table shows the variable type, how much memory it takes to store the value in memory, and what is maximum and minimum value which can be stored in such type of variables:

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647

short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-9223372036854775808 to 9223372036854775807
signed long int	8bytes	same as long int
unsigned long int	8bytes	0 to 18446744073709551615
long long int	8bytes	-(2^63) to (2^63)-1
unsigned long long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	
double	8bytes	
long double	12bytes	
wchar_t	2 or 4 bytes	1 wide character

The size of variables might be different from those shown in the above table, depending on the compiler and the computer you are using.

Following is the example, which will produce correct size of various data types on your computer:

```
#include <iostream>
using namespace std;
int main() {
  cout << "Size of char : " << sizeof(char) << endl;
  cout << "Size of int : " << sizeof(int) << endl;
  cout << "Size of short int : " << sizeof(short int) << endl;
  cout << "Size of long int : " << sizeof(long int) << endl;
  cout << "Size of float : " << sizeof(float) << endl;
  cout << "Size of double : " << sizeof(double) << endl;
```

This example uses **endl**, which inserts a new-line character after every line and << operator is being used to pass multiple values out to the screen. We are also using **sizeof()** function to get size of various data types.
When the above code is compiled and executed, it produces the following result which can vary from machine to machine:

```
Size of char : 1
Size of int : 4
Size of short int : 2
Size of long int : 4
Size of float : 4
Size of double : 8
```

Following is another example:

```
#include <iostream>
#include <limits>
using namespace std;
int main() {
  std::cout << "Int Min " << std::numeric_limits<int>::min() << endl;
  std::cout << "Int Max " << std::numeric_limits<int>::max() << endl;
  std::cout << "Unsigned Int  Min " << std::numeric_limits<unsigned int>::min() << endl;
  std::cout << "Unsigned Int Max " << std::numeric_limits<unsigned int>::max() << endl;
  std::cout << "Long Int Min " << std::numeric_limits<long int>::min() << endl;
  std::cout << "Long Int Max " << std::numeric_limits<long int>::max() << endl;
  std::cout << "Unsigned Long Int Min " << std::numeric_limits<unsigned long int>::min() <<endl;
  std::cout << "Unsigned Long Int Max " << std::numeric_limits<unsigned long int>::max() << endl;
}
```

typedef declaration

You can create a new name for an existing type using typedef. Following is the simple syntax to define a new type using typedef:

typedef type newname;

For example, the following tells the compiler that feet is another name for int:

typedef int feet;

Now, the following declaration is perfectly legal and creates an integer variable called distance:

feet distance;

Functions in C++

A function is a block of code that performs a specific task. Suppose we need to create a program to create a circle and color it. We can create two functions to solve this problem:

1. a function to draw the circle.
2. a function to color the circle.

Dividing a complex problem into smaller chunks makes our program easy to understand and reusable. There are two types of function:

1. Standard Library Functions: Predefined in C++
2. User-defined Function: Created by user

User-Defined Functions

C++ allows the programmer to define their own function. A user-defined function groups code to perform a specific task and that group of code is given a name (identifier).

When the function is invoked from any part of the program, it all executes the codes defined in the body of the function.

The syntax to declare a function is:

```
returnType functionName (parameter1, parameter2,...) {
   // function body
}
```

Here's an example of a function declaration:

```
// function declaration
void greet() {
   cout << "Hello World";
}
```

Here, the name of the function is **greet()** the return type of the function is **void** the empty parentheses mean it doesn't have any parameters the function body is written inside **{}.**

Calling Functions

In the above program, we have declared a function named greet(). To use the greet() function, we need to call it. Here's how we can call the above greet() function:

```
int main() {

   // calling a function
   greet();
}
```

Functions Parameters

As mentioned above, a function can be declared with parameters (arguments). A parameter is a value that is passed when declaring a function.

For example, let us consider the function below:

```
void printNum(int num) {
   cout << num;
}
```

Here, the **int** variable **num** is the function parameter. We pass a value to the function parameter while calling the function:

```
int main() {
   int n = 7;

   // calling the function
   // n is passed to the function as argument
   printNum(n);

   return 0;
}
```

Example: Function with Parameters

```
// program to print a text
#include <iostream>
using namespace std;
// display a number
void displayNum(int n1, float n2) {
   cout << "The int number is " << n1;
   cout << "The double number is " << n2;
}
int main() {
    int num1 = 5;
    double num2 = 5.5;
   // calling the function
   displayNum(num1, num2);

   return 0;
}
```

In the above program, we have used a function that has one int parameter and one double parameter. We then pass **num1** and **num2** as arguments. These values are stored by the function parameters **n1** and **n2** respectively.

Return Statement

In the above programs, we have used void in the function declaration. For example:

```
void displayNumber() {
   // code
}
```

This means the function is not returning any value. It's also possible to return a value from a function. For this, we need to specify the returnType of the function during function declaration.
Then, the return statement can be used to return a value from a function. For example:

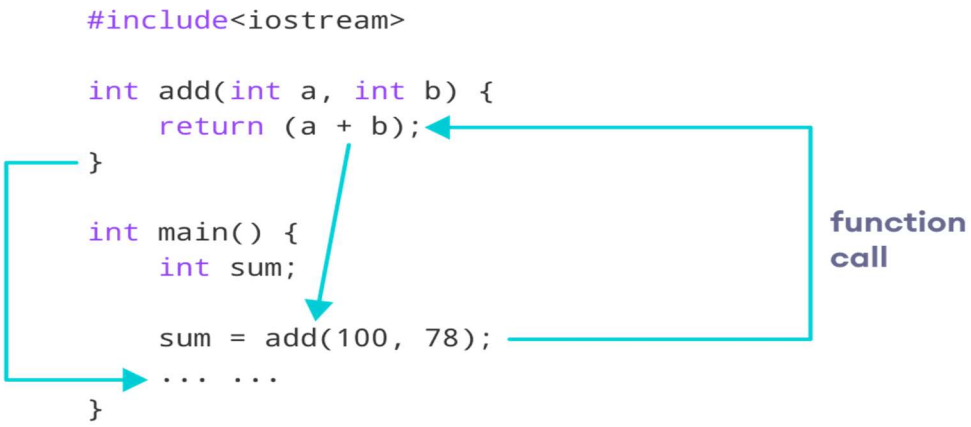
```
int add (int a, int b) {
   return (a + b);
}
```

Here, we have the data type int instead of void. This means that the function returns an int value.
The code return (a + b); returns the sum of the two parameters as the function value. The return statement denotes that the function has ended. Any code after return inside the function is not executed.

Example: Add Two Numbers

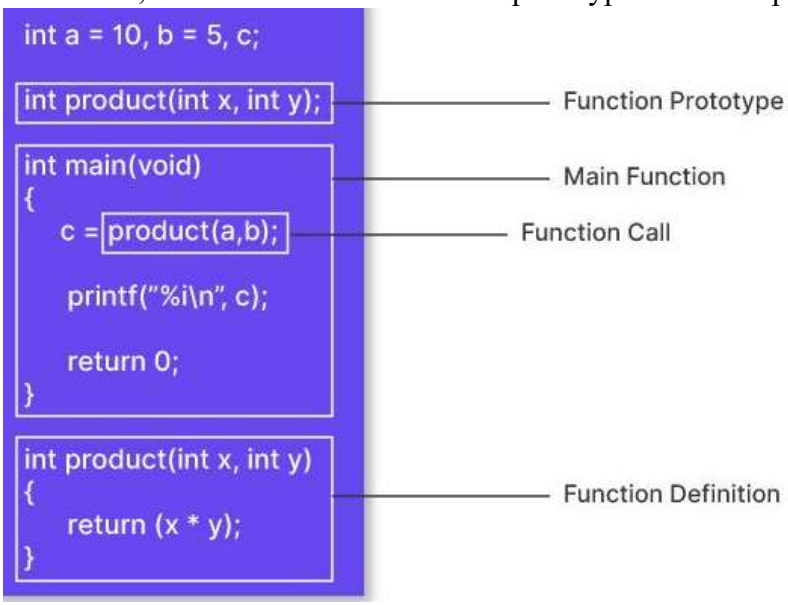
```
// program to add two numbers using a function
#include <iostream>
using namespace std;
// declaring a function
int add(int a, int b) {
   return (a + b);
}
int main() {
   int sum;
   // calling the function and storing
   // the returned value in sum
   sum = add(100, 78);
   cout << "100 + 78 = " << sum << endl;
   return 0;
}
```

In the above program, the add() function is used to find the sum of two numbers. We pass two int literals 100 and 78 while calling the function. We store the returned value of the function in the variable sum, and then we print it.



Function Prototype

In C++, the code of function declaration should be before the function call. However, if we want to define a function after the function call, we need to use the function prototype. For example:



This provides the compiler with information about the function name and its parameters. That's why we can use the code to call a function before the function has been defined. The syntax of a function prototype is:

```
returnType functionName(dataType1, dataType2, ...);
```

Example: C++ Function Prototype

```
// using function definition after main() function
// function prototype is declared before main()
#include <iostream>
using namespace std;
// function prototype
int add(int, int);
int main() {
    int sum;
    // calling the function and storing
    // the returned value in sum
    sum = add(100, 78);
    cout << "100 + 78 = " << sum << endl;
    return 0;
}

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

Functions make the code reusable. We can declare them once and use them multiple times. Functions make the program easier as each small task is divided into a function. Functions increase readability.

C++ Library Functions

Library functions are the built-in functions in C++ programming. Programmers can use library functions by invoking the functions directly; they don't need to write the functions themselves.
Some common library functions in C++ are sqrt(), abs(), isdigit(), etc.
In order to use library functions, we usually need to include the header file in which these library functions are defined. For instance, in order to use mathematical functions such as sqrt() and abs(), we need to include the header file cmath.

Example: C++ Function Prototype

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
   double number, squareRoot;
   number = 25.0;
   // sqrt() is a library function to calculate the square root
   squareRoot = sqrt(number);
   cout << "Square root of " << number << " = " << squareRoot;
   return 0;
}
```

Arrays in C++

An Array is a collection of fixed number of elements of same data type.

1-D Array

1-D Array is a form of array in which elements are arranged in a form of List. To declare a 1D array you need to specify the data type, name and array size.

dataType arrayName [arraySize];

Following is an example of declaration of a 1D array:

int numArray[5];

To access array elements, you use the array name along with the index in subscript operator **"[ ]".**

numArray[0], numArray[1], numArray[2], numArray[3], numArray[4]

Example: Code for 1-D Array

Program to read five numbers, find their sum, and print the numbers in reverse order.

```
#include<iostream>
using namespace std;
int main() {
   int item[5]; // Declare an array 'item' of five components
   int sum = 0;
   int counter;
   cout << "Enter five numbers: ";
   // Input five numbers into the array and calculate the sum
   for (counter = 0; counter < 5; counter++) {
      cin >> item[counter];
      sum = sum + item[counter];
   }
   cout << endl;
   cout << "The sum of the numbers is: " << sum << endl;
   cout << "The numbers in reverse order are: ";
   // Print the numbers in reverse order
   for (counter = 4; counter >= 0; counter--)
      cout << item[counter] << " ";
   cout << endl;
   return 0;
}
```

2-D Array

2-D Array is a collection of fixed collection of elements arranged in rows and columns. To declare a 2D array you need to specify the data type, name and no. of rows and columns.

dataType arrayName [rowSize][columnSize];

Following is an example of declaration of a 2D array:

int numArray[5][5];

To access array element you use the array name along with the row Index and column Index in subscript operator **"[ ][ ]".**

numArray[0][0], numArray[1][1], numArray[2][2], numArray[3][3], numArray[4][4];

Example: Code for 2-D Array

Program to read a 2D array of size 3x3 find the sum for each row, print the sum line by line.

```
#include <iostream>
using namespace std;  int main()
{
int item[3][3];      //Declare an array of size 3x3
 int sum = 0;
int row, col;
cout << "Enter array elements: " << endl;
for (row = 0; row < 3; row++)
{
for (col = 0; col < 3; col++)
{
cin >> item[row][col];
sum = sum + item[row][col];
}
cout << "The sum of row " << i << " : " << sum << endl;
}
cout << endl; return 0;
}
```

Lab Tasks:

1. Write a program that finds the second highest number in a float type array of 20 elements using pointer.

2. Write a program that calculates the sum of all the elements in array using pointers

3. Write a program in C++ to calculate and print the Electricity bill of a given customer. The customer id, name and unit consumed by the user should be taken from the keyboard and display the total amount to pay to the customer. The charges are as follow:

Unit	Charge/Unit
Up to 199	@16.20
200 and above but less than 300	@20.10
300 and above but less than 500	@27.10
500 and above	@35.90

If bill exceeds Rs. 18000 then a surcharge of 15% will be charged on top of the bill.
Test Input:
1001 //Customer ID
James //Customer Name
800 //Units Consumed
Expected Output:
Customer ID :1001
Customer Name: James
Units Consumed :800
Amount Charges @Rs. 35.90 per unit: 28720
Surcharge Amount: 4308
Net Amount Paid by the Customer: 33028.00

4. Write a program that prompts the user to enter the weight of a person in kilograms and outputs the equivalent weight in pounds. Output both the weights rounded to two decimal places. (Note that 1 kilogram = 2.2 pounds.) Format your output with two decimal places.

5. A movie in a local theater is in great demand. To help a local charity, the theater owner has decided to donate to the charity a portion of the gross amount generated from the movie. This example designs and implements a program that prompts the user to input the movie name, adult ticket price, child ticket price, number of adult tickets sold, number of child tickets sold, and percentage of the gross amount to be donated to the charity. The output of the program is as follows.

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Movie Name: ..... Journey to Mars
Number of Tickets Sold: ..... 2650
Gross Amount: ..... $ 9150.00
Percentage of Gross Amount Donated: 10.00%
Amount Donated: ..... $ 915.00
Net Sale: ..... $ 8235.00
```

Note that the strings, such as "Movie Name:" , in the first column are left-justified, the numbers in the right column are right-justified, and the decimal numbers are output with two decimal places.

Input: The input to the program consists of the movie name, adult ticket price, child ticket price, number of adult tickets sold, number of child tickets sold, and percentage of the gross amount to be donated to the charity.

Output: The output is as shown above.

6. Write a program that reads a student name followed by five test scores. The program should output the student's name, the five test scores, and the average test score. Output the average test score with two decimal places.

Input:
Andrew Miller 87.50 89 65.75 37 98.50

Output:
Student name: Andrew Miller
Test scores: 87.50 89.00 65.75 37.00 98.50
Average test score: 75.55