Higher Diploma in Information Technology

Introduction to Programming (C++)

Year 1 Semester 1 – 2023

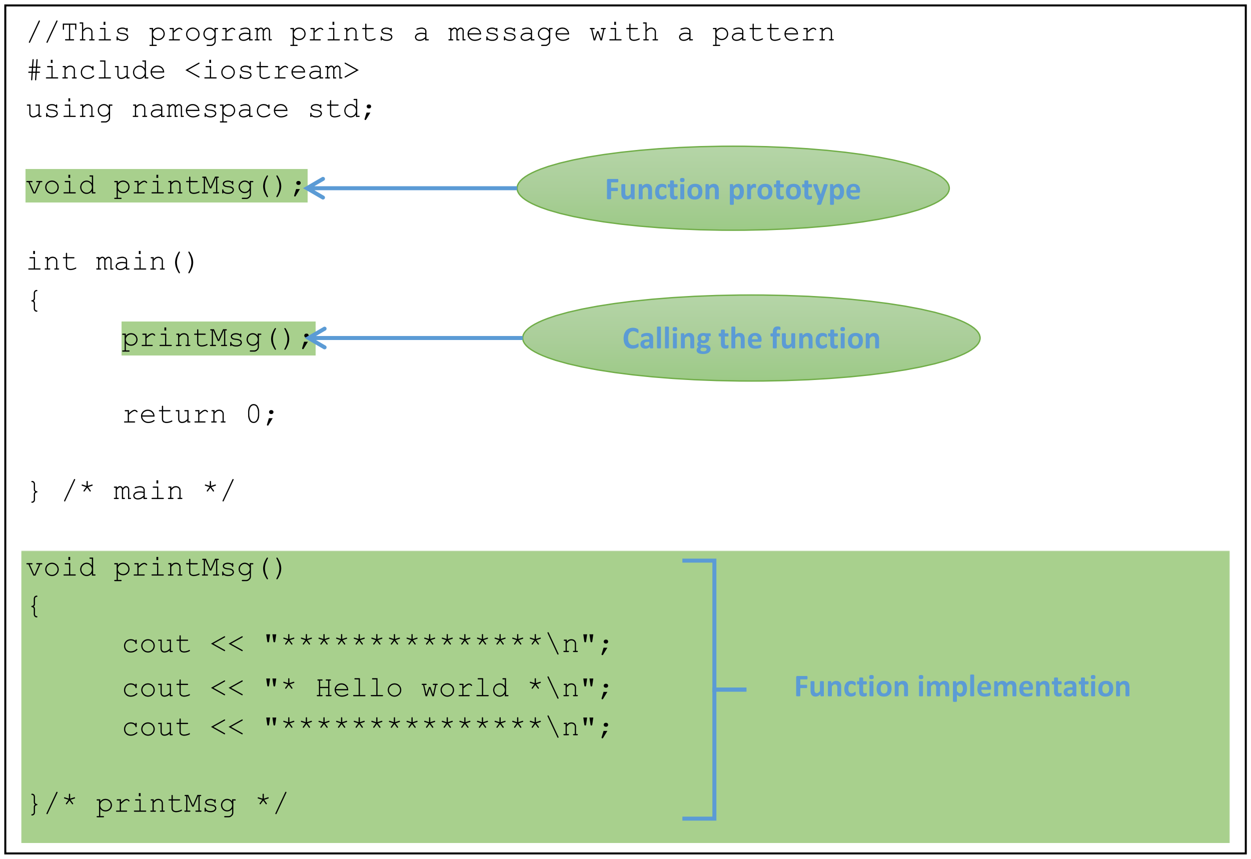
Worksheet 06

Intended Learning Outcomes:

At the end of the class the students should be able to:

- Break the initial problem in to sub problems.
- Implement and use functions in C++ programs.
- Identify the difference between pass by value and pass by reference parameter passing methods.

Exercise 01: Create a simple C++ program as **Ex01.cpp** and observe the function implementation**.**



```cpp
//This program prints a message with a pattern
#include <iostream>
using namespace std;

void printMsg();

int main()
{
	printMsg();

	return 0;

} /* main */

void printMsg()
{
	cout << "***************\n";
	cout << "* Hello world *\n";
	cout << "***************\n";

}/* printMsg */
```

Exercise 02: Write a program as **Ex02.cpp** to implement a function called `drawLines()` to draw the below figure. And call the `drawLines()` function in your main program to draw the rectangle.

```
**********
**********
**********
**********
```

NOTE: To implement the function you can use repetition structures such as for-loop structure.

Exercise 03: Write a program called **Ex03.cpp** to implement another function called drawLinesWithRow() to pass the no of lines(rows) as a parameter.

E.g.: `void drawLinesWithRow(int rows);`

If 3 is passed as the number of rows, the function should print,

```
**********
**********
**********
```

Call the function created in your main program and draw two rectangles with 6 and 12 rows. Use repetition structures such as for-loop structure to implement the pattern.

Exercise 04: Write another program as **Ex04.cpp** to implement a function called `drawLinesWithRowCol()` to pass the no of rows and columns as parameters.

E.g.: `void drawLinesWithRowCol (int rows, int cols);`

If 3 is passed as the number of rows and 5 as the number of columns, the function should print,

```
*****
*****
*****
```

Exercise 05:

A program is required to calculate and print the net salary of an employee depending on the basic salary and the commission. The user enters the position and the sales done as user input to calculate the basic salary and commission, respectively. ***Hint: use the global variables.***

Consider the following to write the functions:

A. Implement a function called **readData()** to read the emp no, emp name, executive position and no of sales.

B. Implement a function called **findBasicSalary()** to find the basic salary scale of the employee. A company offers the following basic salary scales for two types of marketing executive positions.

Position	Basic Salary
1	25000
2	50000

If an invalid executive position is entered. display an error message.

Hint: Use nested-if statements.

C. Implement a function called **findRate()** to find the commission rate based on the sales are done. The commission rates are given below. ➔

Hint: Use nested-if statements.

Sales	Commission Rate
6000 or above	20%
4000-5999	15%
3000-3999	10%
Below 3000	5%

D. Implement a function called **calculateCommission()** to calculate the commission of the employee.

Commission=basic salary * Commission Rate

E. Implement a function called **calculateNetSalary()** to calculate the net salary of the employee.

Net salary = basicsalary+ commission

F. Implement a function called **printDetails()** to print the employee details. (Details include emp no, emp name, net salary)

G. Implement the main method to call the respective modules to perform the following tasks.

- Read the inputs from the user.
- Find the basic salary.
- Find the commission rate.
- Calculate the commission.
- Calculate the net salary.
- Print the employee details.

```
Enter Employee No: E_111
Enter Employee Name: Jagath Silva
Enter Position: 1
Enter no of Sales :3500
The Employee No is E_111
The Employee name is Jagath Silva
The Net salary is 27500
```

Exercise 06: Now create a simple C++ to observe the parameter passing – Pass by value. Save the program with the names of **Ex06.cpp.**

```
//An Example of pass by value parameter passing
//This program calculate the sum of two integers

#include <iostream>
using namespace std;

int calSum(int a, int b);

int main()
{
      int x, y, ans;

      cout <<"Enter the value of x:";
      cin >> x;
      cout <<"Enter the value of y:";
      cin >> y;

      ans = calSum(x, y);

      cout <<"The sum of "<<x<<" and "<<y<<" is "<<ans<< endl;

      return 0;

} /* main */

int calSum(int a, int b)
{
      int sum = a + b;
      a++;
      cout << "The new value of a is " << a << endl;
      return sum;
}/* calSum */
```

Pass by value is a method of passing information to a function whereby the parameter receives a copy of the value of the argument.

Variable *a* receives a copy of the value of the argument *x*. Therefore any changes that the function makes to the parameter (*a*) when pass by value is used are made to the copy and not to the original argument (*x*).

Exercise 07: Write a function called `circleArea()` to find the area of a given circle when the radius is given as a parameter. Write another function called `rectangleArea()` to find the area of a rectangle when length and width are given as parameters. Function prototypes are given below.

```
float circleArea(float radius);
float rectangleArea(float length, float width);
```

Save your program as **Ex07.cpp**

Exercise 08: Now create a simple C++ to observe the parameter passing – Pass by reference.

Save your program as **Ex08.cpp**

```
//An Example of pass by reference parameter passing
//This program calculate the average of two integers

#include <iostream>
using namespace std;

void calAvg(int a, int b, float &avg);

int main()
{
      int x, y;
      float average = 0.0;

      cout <<"Enter the value of x:";
      cin >> x;
      cout <<"Enter the value of y:";
      cin >> y;

      calAvg(x, y, average);

      cout <<"The average is "<< average << endl;

      return 0;

} /* main */

void calAvg(int a, int b, float &avg)
{
      int sum = a + b;
      avg = sum/2.0;

}/* calAvg */
```

Pass by reference or call by reference is the technique of making a variable accessible to a function by passing its address.

The function has direct access to the argument (*average*), through its address; the function may modify the argument.