UEE1303 S18: Object-Oriented Programming

Advances Topic on Functions

What you will learn from Lab 1

In this laboratory session you will:

Learn how to use the inline function, function overloading and function template

LAB 1-1: INLINE FUNCTIONS

✓ The following two examples are used to demonstrate the difference between ***inline*** functions and ***define*** macro.

```
// lab1-1-1.cpp
#include <iostream>
#define PI 3.14159
#define circleArea(x) (PI*(x)*(x))

int main()
{
	std::cout << "circleArea(4.3) = " << circleArea(4.3) << std::endl;
	std::cout << "circleArea(4) = " << circleArea(4) << std::endl;

	return 0;
}
```

```
// lab1-1-2.cpp
#include <iostream>
#define PI 3.14159
inline double circleArea(int x) {return PI*x*x;}

int main()
{
	std::cout << "circleArea(4.3) = " << circleArea(4.3) << std::endl;
	std::cout << "circleArea(4) = " << circleArea(4) << std::endl;

	return 0;
}
```

➢ Although define macro results in the answer as you expect, inline functions follow all the protocols of type safety enforced on normal functions. When you compile the lab1-1-2.cpp file with inline functions, your compiler will show the warning message about erroneous type conversion.

LAB 1-2: FUNCTION OVERLOADING

✓ Execute the following program lab1-2.

```
//File: lab1-2.cpp
#include <iostream>
using namespace std;
double avg(double, double, double);
double avg(double, double);
int main()
{
   cout << "The average of 2.0, 2.5, and 3.0 is " << avg(2.0, 2.5, 3.0) << endl;
   cout << "The average of 4.5 and 5.5 is " << avg(4.5, 5.5) << endl;
   return 0;
}
double avg(double n1, double n2, double n3)
{
   return ((n1 + n2 + n3) / 3.0);
}
double avg(double n1, double n2)
{
   return ((n1 + n2) / 2.0);
}
```

✓ Modify the program lab1-2.cpp as lab1-2-1.cpp

```
// lab1-1-2.cpp
#include <iostream>
#define PI 3.14159
inline double circleArea(int x) {return PI*x*x;}

int main()
{
      std::cout << "circleArea(4.3) = " << circleArea(4.3) << std::endl;
      std::cout << "circleArea(4) = " << circleArea(4) << std::endl;

      return 0;
}
```

- ➢ Note that the functions can only be overloaded by different arguments but cannot be overloaded by different return types.
- ➢ ceil is a round up value which is used to return the smallest integral value that is not less than x.
- ➢ Please modify the program to obtain the correct results.

EXERCISE 1-1: THE AREA OF THE CIRCLE

✓ Write a C++ program that prompts the user for the radius of a circle, then calls inline function circleArea to calculate the area of that circle

```
>./ex1-1
Enter the radius of the circle: 4
The area of the circle is 50.2654
>
```

- ✓ Not that PI is 3.1415 in this exercise.

Exercise 1-2: Complex Arithmetic

- ✓ Please write a C++ program to perform the arithmetic operations for complex numbers. You have to read two complex numbers and output the arithmetic results to the console.

```
> ./ex1-2
First complex number: 1.5+6i
Second complex number: -2-10i
==============================
The output results:
A + B = -0.500-4.000i
A – B = 3.500+16.000i
A * B = 57.000-27.000i
A / B = -0.606+0.029i
```

- ✓ You have to use the following data structure Cplex and write two functions: complexOperation() and printComplex().
- ✓ The ex1-2.cpp has the content as: (some headers should be added)

```
// ex1-2.cpp
typedef struct {
      double real;
      double image;
} Cplex;

int main(int argc, char *argv[])
{
      Cplex a, b;
      // promotes the user to input data
      ………
      // store the results of diff. operation
   Cplex results[4];
   results[0] = complexOperation(a, b, '+');
   results[1] = complexOperation(a, b, '-');
   results[2] = complexOperation(a, b, '*');
   results[3] = complexOperation(a, b, '/');
   printComplex(results);

   return 0;
}
```