CMPE – 211

Preliminary Work (Pre-Lab Activity)

Laboratory Experiment #5

Textbook Material:	Chapters 1- 9	*pp.1-230*	[see Laboratory Experiments #3 ÷ #4]
	Chapter 10	*«Classes»*	*pp.232-255*

• • • TASK 1

Write (complete sections), Compile and Execute (Test) a variant of C++ program whose template («skeleton») is given below.

```
#include<iostream>
#include<cstring>
using namespace std;
class CObj {                          // definition of the class starts here
        char *Name;
        public:                       // public section specifies interface of the class
// two constructors are used - they are overloaded functions (note that their signatures are different)
                CObj(char*);          // constructor with parameters (1)
                CObj();               // (constructor) - we want to form name by default (2)
                ~CObj();              // destructor of the class
                void WhatIsYourName(void);     // member function of the class
};
// end of the class definition

CObj::CObj(char *YourName)            // definition of the constructor (1)
{
        //  write definition of the constructor, which performs the following work:
        //  1. allocates memory dynamically for the array (function's parameter)
        //  2. initializes dynamic array with a string passed
        //  3. displays the output: I was born under the name <YourName>
}
Cobj::CObj(void)                      // definition of the constructor (2)
{
        //  write definition of the constructor, which performs the following work:
        //  1. it does exactly the same as constructor (1) does, but for the string "Noname"
}
void CObj::WhatIsYourName(void)       // definition of member function
{
        //  function displays the Name (string)
}
CObj::~CObj(void)                     // definition of the destructor
{
        //  function releases (frees) previously allocated dynamic memory
}
int main()                            // the body of the function main( ) is already written
{
        CObj First, Second("Yanick"); // two instances (objects) of the class CObj
        First.WhatIsYourName();       // call to member function for the object First
        Second.WhatIsYourName();      // call to member function for the object Second
        return 0;
}
```

Finally, after completing the code and executing the resultant program, it should produce the following output:

```
I was born under the name Noname
I was born under the name Yanick
```

My name is Noname
My name is Yanick
The memory is released…
The memory is released…

● ● ● TASK 2

Write, Compile and Execute (Test) a C++ program that uses class **Complex** for performing **arithmetic with complex numbers** (see Calculus I, II courses).

Complex numbers have the form realPart + imaginaryPart · i (e.g. $-5 + 2 \cdot i$), where i is $\sqrt{-1}$, $i^2 = -1$. Use **double** variables to represent the private data of the class. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializers are provided. Provide public member functions for each of the following:

1. adding two **Complex** numbers (two objects of the class **Complex**) – the real parts are added together, and the imaginary parts added together (e.g. $(1.7 + 2 \cdot i) + (0.6 - 1.4 \cdot i)$ gives $2.3 + 0.6 \cdot i$),
2. multiplying two **Complex** numbers (e.g. $(2 + 3 \cdot i) \cdot (3 + 4 \cdot i)$ is performed in an «each-by-each» manner, that is $6 + 8 \cdot i + 9 \cdot i - 12 = -6 + 17 \cdot i$),
3. printing **Complex** numbers in the form (a, b), where a is a *real part* and b is the *imaginary part* of a complex number.

Reference:
a) An Introduction to Complex Numbers (http://www.ping.be/~ping1339/complget.htm)
b) Wolfram Research (http://mathworld.wolfram.com/ComplexNumber.html)
c) Complex Numbers (http://www.mathaid.com/products/ComplxNumbrs/full/contents.html)

● ● ● TASK 3

Write down a user-defined type (class) **Triangle** that represents triangles in Cartesian coordinate system. Check whether your definition works properly (prepare a complete C++ program that tests the class). Some specifications to follow while preparing solution of the task are given below.

(-a) a triangle is made up of three edges and three corners. Each corner should be represented as a point in the Cartesian coordinate system (i.e. as a 2-value pair **(x,y)**). The base edge is located on the axis **x**.
HINT: Coordinates can be stored in a a two-dimensional array (3 rows and 2 columns) of **double**s declared as a private data member of the corresponding class,

(-b) there should be a member function **changeBase( )** to change the «base» (axis **x**) of the triangle,

(-c) there should be a member function **changeHeight( )** to change the «height» (axis **y**) of the triangle at any time,

(-d) another member function **computeArea( )** is the one to compute the area of the triangle,

(-e) include function **equilateral( )** that determines if the triangle is equilateral,

(-f) provided class' constructor creates a triangle with a base length of 2 cm and the height of 1 cm – its left bottom stands at the origin (point with coordinates **(0,0)**),

(-g) class' destructor simply destroys the triangle by assigning **(0,0)** to corner point pairs.

● ● ● TASK 4 (see problem 10.5 in the textbook)

Implement a Matrix class for N-by-M matrices having N rows and M columns. In the default constructor, a 2-by-2 matrix is defined and its entries are initialized to zero. Define another constructor that allows dynamic allocation of N-by-M matrices where the initial values are left undefined. The destructor should de-allocate the matrix. The number of rows and columns should be kept in two private data members named as **rowNum** and **colNum**. Define an access function **Display()** to display the matrix entries. Similarly, define a mutator function **Set(int i,int j,double val)** to set the value of **ith** row and **jth** column to **val**. Write, Compile and Execute a C++ program that makes use of the class designed.

● ● ● Appendix

- Check Review Questions at the end of textbook's Chapter 10 (page 249) and review pointers, dynamic memory allocation in C++ (using operators **new** and **delete**), basic class concepts (definition, data members, member functions, constructors/destructors)
- Use debugging facilities of the Visual C++ compiler while writing programs for TASKS 1-4 (follow tutorial «Dive Into Microsoft Visual C++ 6» by Deitel & Associates)

• • • Sources

- *John R. Hubbard*. Schaum's Outline of Programming with C++, 2nd edition, McGraw-Hill, 422 p., 2000
- *Harvey M. Deitel, Paul J. Deitel*. C++ How To Program, 4th edition, Prentice Hall, 1320 p., 2002
- Examples discussed in the training seminar «Programming in C++» (*Dr.D.Maksin*, Training Center «Specialist», 2002)
- Introduction to Classes (Lesson 12), http://www.cprogramming.com/tutorial/lesson12.html
- C/C++: Classes, Members and Methods (Lesson 17), http://cplus.about.com/library/weekly/aa070602a.htm
- C/C++: Constructors and Desctructors (Lesson 18), http://cplus.about.com/library/weekly/aa072302a.htm
- *Frank B. Brokken*. C++ Annotations (ver.5.1.0b), http://ari.cankaya.edu.tr/~guvenc/ceng112/cplusplus.html
  PDF version (zipped file, 3.69Mb) of the document can be found at
  ftp://ftp.rug.nl/contrib/frank/documents/cplusplus.annotations/cplusplus.pdf.zip
- *Peter Müller*. Introduction to Object-Oriented Proramming Using C++, 1997
  http://www.gnacademy.org/text/cc/Tutorial/tutorial.html
- *Joseph Gil*. Classes in a Nutshell (OOP Lectures), Uppsala University, 1998
  http://www.csd.uu.se/kurs/oop/ht98/Lectures/D1/html/Classes_in_a_Nut-Shell/index.htm