CMPE – 211

Preliminary Work (Pre-Lab Activity)

Laboratory Experiment #6

Textbook Material:	Chapters 1- 9	*pp.1-230*	[see Laboratory Experiments #3 ÷ #5]
	Chapter 10	*«Classes»*	*pp.232-255*
	Chapter 11	*«Overloading Operators»*	*pp.256-259* *(beginning of the chapter)*

• • • TASK 1

Read corresponding chapter extracts and Check in practice concepts (review questions/selected problems) of using

(A) Standard C++ **string** class (type) (Chapter 9, pp. 213-232 of the textbook),

(B) **static** data members and **static** function members (Chapter 10, pp. 245-248 of the textbook).

For example, consider the following C++ code[1]:

```
#include <iostream>
#include <cstring>
using namespace std;
class MyString  {                       // definition of the class MyString (user-defined type)
      const char * s;                   // pointer to text string
      int lens;                         // length of the string
      int my_number;                    // "personal" number
      static int counter;               // pay attention to the way this variable (static data member)
                                        // is used in the program
   public:
      void SetSense(const char * str)  {    // definition of the public function member
           my_number = ++ counter;
           s = str;
           lens = strlen(s);                // C-strings (arrays of characters) are used
      }
      void TellAboutYourself( ) const;      // definition of the function is given below
      static int HowMany( )  { return counter; }        // function works as a counter
                                                        // (see code below)
};    // end of the class MyString definition
int MyString :: counter = 0;            // compiler allocates counter in data segment of the memory
int main( )
{
      cout << "In total " << MyString :: HowMany( ) << " strings are created";
      cout << endl;
      MyString S1, S2, S3;              // three objects (instances) of the class MyString are created
      S1.SetSense("Good morning, world!");          // function SetSense( ) is called for object S1
      S2.SetSense("Long live programming in C++!");
      S3.SetSense("We like to make experiments...");
      cout << "In total " << MyString :: HowMany( ) << " strings are created" << endl;
      cout << "Why a call to S1.HowMany( ) returns " << S1.HowMany( ) << "?" << endl;
      S1.TellAboutYourself( );          // same function is called for three different objects
      S2.TellAboutYourself( );
      S3.TellAboutYourself( );
      return 0;
}     // end of the function main( )
void MyString :: TellAboutYourself( ) const       // definition of the function TellAboutYourself( )
{
      const char * sfp = "I am a very nice string!\nMy personal number is ";
      cout << sfp << my_number << endl
           << "I keep the text: " << s << endl
           << "Its length is " << lens << " characters. That's all!" << endl;
      return;
}
```

[1] Based on the example from the book «From C to C++» by D.Rassokhine (M., EDEL, 1993, 128 p.)

Compile and Execute a given C++ program – are there any errors? Make necessary corrections, if needed, and explain the results of code execution. CHECK: Is it possible to use C++ **string** type instead of C-strings (arrays of characters terminated by nul character '\0') utilized in the current version of the program? Rewrite the code using type **string** wherever possible and examine it.

• • • TASK 2

Consider carefully the following C++ code:

```
#include<iostream>
#include<cstring>
using namespace std;
class MY {
      char* Name;     // private member of the class MY
      public:
            MY( )
            {
                  cout << "Inside DEFAULT constructor...\n";  // reminder message is printed
                  int i = strlen("Noname");
                  // variable i is initialized with a length of the string "Noname"
                  Name = new char[i+1];          // 1D-array of characters is allocated dynamically
                  strcpy(Name, "Noname");        // allocated array array is initialized with string
            }
            MY(const MY&);          // copy constructor of the class is declared (its definition
                                    // is provided outside the class)
            void WhatIsYourName(void)
            {
                  cout << "My name: " << Name << '\n';
            }
            ~MY( )          // definition of the class' destructor (deallocation of dynamic memory)
            {
                  cout << "Inside destructor...\n";    // reminder message is printed
                  delete[ ] Name;
            }
};
      // we define a function that returns the object of the class MY – this function is NOT a member
      // of the class MY
MY Funct(MY& Obj)    // object is passed to the function by reference
{
      cout << "Inside function Funct( )...\n";     // reminder message is printed
      MY Obj1 = Obj;
      return Obj1;
}
MY :: MY(const MY& Obj)              // definition of the copy constructor
{
      cout << "Inside COPY constructor...\n";      // reminder message is printed
      strcpy(Name = new char[strlen(Obj.Name)+1], Obj.Name);
}
int main( )     // definition of the function main( )
{
      MY First;
      First.WhatIsYourName( );
      MY Second = Funct(First);
      MY Third(First);
      Second.WhatIsYourName( );
      return 0;
}
```

Compile and Execute a given C++ program – are there any errors? Make necessary corrections, if needed, and explain the results of the code execution. CHECK: If the program displays several lines of the output, carefully examine each of them. Pay **much attention** to those statements that cause COPY constructor to be called. CHECK: Under which circumstances COPY constructor is called automatically?

CHECK: What happens if prototype of the function **Funct( )** is changed to **MY Funct(MY Obj);**? What changes should be done in the rest of the code, and what is the effect of such code modifications to the output of the program?

••• TASK 3

Implement a **OneDigit** class (integers from the closed interval 0..9). Include a default and copy constructors, a Boolean function **oddeven( )** to check whether a one-digit integer number is odd or even, get-function **GetOneDigit( )** to access class data, and a **print( )** function. The only data member is the integer variable **num** used for storing numbers under consideration. Write, Compile and Execute a complete C++ program that makes use of the class designed.

■■■ OPTIONAL TASK: Using class **OneDigit** Write, Compile and Execute a complete C++ program that allows representation of two-digit integer numbers. For example, if the first digit is **a** (**a** ∈ [1,9]) and the second one is **b** (**b** ∈ [0,9]), then the result number is displayed as **ab**; if **a** is 0, then number is shown as just **b** (two-digit number is «constructed» as $a \cdot 10 + b$).

••• Appendix

- Check Review Questions at the end of textbook's Chapter 10 (page 249 – problems 10.1÷10.9 imply implementation of classes; consider solutions of these problems (pp. 251-255) and verify them practically on computer). "Refresh" such topics as pointers, dynamic memory allocation in C++ (using operators **new** and **delete**), basic class concepts (definition, data members, member functions, default and copy constructors, destructor)
- Use debugging facilities of the Visual C++ compiler while writing programs for TASKS 1-3 (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
- C/C++: Copy Constructors (Lesson 20), http://cplus.about.com/library/weekly/aa072802a.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