AST21105 Object-Oriented Programming & Design

Lab 3 – Arrays and Functions

A. Submission Details

In this lab, you are required to submit **ONE** C++ program to solve the given problem shown in the section "Master Mind". To make the program implementation easier, you are suggested to write your program using Visual Studio .NET instead of doing it directly using paper & pencil. After you have completed the implementation, submit your program file (i.e. MasterMind.cpp for this lab) using the electronic "drop-box" in Canvas, **One week** after your lab section is conducted. For details, please refer to the following:

Tuesday sections	*by*	*19:00 on 2 February 2016 (Tuesday)*
Wednesday sections	*by*	*19:00 on 3 February 2016 (Wednesday)*

You are reminded to double-check your solution to verify everything is correct before submission. You are also required to put your name and student ID at the beginning of your source file.

Important: You are only able to submit your work once.

B. Objective

The objective of this lab is to help you get familiar with arrays and functions (i.e. a construct similar to methods in Java) that you have learned in Java last year. Also, you will be introduced to the difference between Java and C++ so as to speed you up your learning process of C++.

The first part of this lab is meant to be a simple review of array and function declarations using Microsoft Visual Studio .NET. The second part of this lab is practical task that allow you to use all the techniques introduced (i.e. arrays and functions).

C. Review

1. Declaration of arrays

   An array is a collection of variables that are of the same type. Possible types used are:

Integer data types	short, int, long
Floating point data types	float, double
Character type	char
Boolean type	bool
User-defined type	All other types

Comparing to Java, C++ offers the flexibility to declare an array ***using new*** or ***not using new*** keyword (Recall that array in Java has to be declared using new keyword.). In this lab, we only focus on declaring array ***not using new***.

•Syntax:

// Declare an array named <name of variable> of size <size> in type <type>
<type> <name of variable>[<size>];

where <type> could be in one of the following types:
short, int, long, float, double, char, bool or user-defined data type

Important note:
- <size> has to be a constant, it cannot be a variable.
- The initial values of the array are ***garbage***

(Again, this is another feature that is different from Java. Recall, Java automatically initializes an array of int type to 0, array of double type to 0.0, array of boolean type to false, but this is not the case in C++.)

// Declare an array named <name of variable> in type <type>, with initialize values
<type> <name of variable>[ ] = { <iv1>, <iv2>, …, <ivn> };

where <type> could be in one of the following types:
short, int, long, float, double, char, bool or user-defined data type
where <ivn> is the initial value of element n

Note: The [ ] in this case can be either made empty or an integer refers to the number of initialize values

•Examples:

int arr1[10];   // Declare an array of 10 elements in int type

double arr2[20];   // Declare an array of 20 elements in double type

const int SIZE = 30;

char arr3[SIZE]; // Declare an array with constant SIZE = 30 in char type

int arr4[ ] = { 1, 2, 3, 4, 5 }; // Declare an array of 5 elements initialized to 1, 2, 3, 4, 5

Questions:

(1) Is the statement below a valid C++ statement?

int arr5[5] = { 1, 2, 3, 4, 5, 6 };

Answer:

No. It is invalid. This refers to a “too many initializers” error (a compilation error). The number of initial values (6 in this example) exceeds the number of elements declared (5 in our case).

(2) Is the following statement valid?

int n=5;

cons tint SIZE = n;

int arr6[SIZE];

Answer:

No. It is invalid. The size of the array cannot be a constant initialized by a variable n. Since a variable can be used to store a user input obtained from the keyboard and this is not allowed to be used as the size of an array.

The following example describes the idea:

int n;

cin>>n;

const int SIZE = n;

int arr6[SIZE];

Invalid :(

2. Declaration and definition of user-defined function

Recall, a large software application requires hundreds of thousands, or even millions lines of code. It is impractical to write a single piece of code in main for such as large application. Therefore, modular design principle is introduced, to put a small, but meaningful, piece of code together. Other parts of a program can use this piece of code once or several times. This is so called "**code reuse**".

A crucial part of all such schemes is the **use of functions**.

- Syntax:

```
// Declare a user-defined function
<return-type> <function-name>(<parameter-list>)
{
    // function body (useful code)
}
```

where <return-type> refers to the type of data returned back to the function caller and could be in one of the following types:
short, int, long, float, double, char, bool, void or user-defined data type
<function-name> is the name of the function, and <parameter-list> is a list of comma-separated variables with their types

```
// Function call
<function-name>( <variable-list> );
```

<variable-list> is a list of comma-separated variables.

- Examples:

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

// A function which returns the maximum of two integers
int max( int a, int b ) {               /* PASS-BY-VALUE */
    return (a > b) ? a : b;
}

int main() {
    cout << "Enter two integers "; int v1, v2;
    cin >> v1 >> v2;
    int maxval = max(v1,v2);
    cout << "Max value: " << maxval; system("pause");
    return 0;
}
```

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

// Swapping two integers
void swap1( int a, int b ) { /* PASS-BY-VALUE */
    int temp = a;
    a = b;
    b = temp;
}
void swap2( int& a, int& b ) {   /* PASS-BY-REFERENCE */
    int temp = a;
    a = b;
    b = temp;
}
```

```
int main() {
    int x = 10, y = 20;
    cout << "Before swap1 - (x,y)="<< x <<","<< y <<endl;
    swap1( x, y );
    cout << "After swap1 - (x,y)="<< x << ", " << y << endl;
    cout << "Before swap2 - (x,y)="<< x << ", " << y << endl;
    swap2( x, y );
    cout << "After swap2 - (x,y)="<< x << ", " << y << endl;
    system("pause");
    return 0;
}
```

Questions:

(1) Which swapping function, swap1 or swap2 does the job correctly?

Answer: swap2 is the correct one to do the swapping of two integers.

(2) What is the difference between pass by value and pass by reference?

Answer:

<u>Pass-by-value:</u>

(i) The value of the original variable is copied to the formal parameter.
(ii) Changes to the value of the formal parameter do not affect the original variable.
(iii) Even though the values of parameter a and b are changed, the corresponding values in variable x and y do not change.

<u>Pass-by-reference:</u>

(i) Pass-by-reference does not copy the value of the actual parameters (i.e. x, y in our example) to the formal parameters (a, b in our example)
(ii) Instead, the formal parameters refer to the actual parameters by some mechanism.
(iii) Any modification of the formal parameters directly changes the value of the actual parameters, i.e. if the values of parameter a and b are changed, the corresponding values in variable x and y are changed as well.
(iv) To pass the parameters by reference, we have to ***add a symbol &*** after the type of the formal parameters.

D. Master Mind

In this part, you are required to:

i) Create a New Project and give your project the name Lab3b.

ii) Add a source file to your project, called MasterMind.cpp

iii) Write a C++ program to implement the game MasterMind. Your code should perform the following.

i. Randomly place ***different values*** to an array of 4 integer values. The values placed should be in the range 1 to 6.

ii. Allow a user to guess the number and computer should respond to the guess with clues.

- A "O" symbol means "right value, right position". That is one of your digits has the right value in the right position.
- A "#" symbol means "right value, wrong position". That is one of your digits has the right value, but is in the wrong position.
- If the computer responds with no pegs, that means everything is wrong.

(Note: You shouldn't give any cues to user about their input integers. What you need to do is to let him/her know how many integers are "right value, right position" and how many are in "right value, but in wrong position".
REQUIREMENT: "O" symbol has to be placed before any "#" symbol)

iii. The output of the game should look like below:

```
MasterMind (For checking: 6 4 3 1)


Enter four digits (1-6) separated by a space
--------------------------------------------

 Round 1:
  Enter Guess:   1   2   3   4

                 O   #   #
--------------------------------------------

 Round 2:
  Enter Guess:   4  6    3  1
                 O  O    #  #
--------------------------------------------
...
```

```
--------------------------------------------
 Round 8:
  Enter Guess:   6  4   3   1
                 O  O   O   O
--------------------------------------------
Congratulations! You win the game in 8 steps
```

The input which is underlined is the user’s input to a question.

iv) Compile your program and test it by executing your program.

Hints: In order to help you out for this lab task, the following skeleton code is given for you to get started.

```
 #include <iostream>
 #include <ctime>
 using namespace std;
 //     Write your generateQuestionAndAnswer function here
 //     ...
 //     Write your answerCorrect function here
 //     ...
void quiz()
{
    int guess;
    int number1, number2, answer; bool
    flag = true;

    do{
        if(flag){
           // Pass two variables, number1 and number2,
           // to function "generateQuestionAndAnswer"
           // The function will return back the answer of the
           // remainder.
           // Hint: The variables should be PASSED BY REFERENCE
           answer = generateQuestionAndAnswer(number1, number2);
           cout << "How much is the remainder of " << number1 <<"divided by " << number2
           << "?" << endl;
        }
        cout << "Enter your answer (-1 to exit): ";
        cin >> guess;
        if( guess == -1 )
          break;

        // Pass two variables, guess and answer,
        // to function "answerCorrect".
        // The function will return a boolean value true
        // if guess is equal to answer,
        // otherwise, a boolean value false should be returned
        flag = answerCorrect( guess, answer );
```

```
       if(!flag)
           cout << "No. Please try again." << endl;
       else
           cout << "Very good!" << endl;
   }while( guess != -1 );
}
int main()
{
    srand((unsigned int)time(NULL));
    quiz();
    system("pause");
    return 0;
}
```

Marking Scheme:

Graded items	Weighting
1. Correctness of program (i.e. whether your code is implemented in a way according to the requirements as specified.)	60%
2. Indentation	30%
3. Documentation (with reasonable amount of comments embedded in the code to enhance the readability.)	10%
	100%

Program Submission Checklist

Before submitting your work, please check the following items to see you have done a decent job.

Items to be checked	☑ / ☒
1. Did I put my name and student ID at the beginning of all the source files?	☐
2. Did I put reasonable amount of comments to describe my program?	☐
3. Are they all in .cpp extension and named according to the specification?	☐
4. Have I checked that all the submitted code are compliable and run without any errors?	☐
5. Did I zip my source files using Winzip / zip provided by Microsoft Windows? Also, did I check the zip file and see if it could be opened? ***(Only applicable if the work has to be submitted in zip format.)***	☐
6. Did I submit my lab assignment to Canvas?	☐

-End-