Software Engineering
Lab 2 Report

Emre Ozan Alkan
{emreozanalkan@gmail.com}
MSCV-5

19 October 2013

1 Preliminaries

The code given had some problems and were not able to compile. Like in ".h" file globally defined variable in ".h" file was not have "extern" keyword. Also the functions in ".cpp" file doesn't have any return type. We rewrited the code and was able to compile code.

Listing 1: Lab2.h

```

#ifndef __SELab2_1__Lab2__
#define __SELab2_1__Lab2__

#include <iostream>

using namespace std;

// Our dumpvar for test
extern int dumvar;

int MyFunction1(int, int);

void MyFunction2(float);

#endif /* defined(__SELab2_1__Lab2__) */
```

Listing 2: Lab2.cpp

```

#include "Lab2.h"

using namespace std;

// Our dumpvar initization
int dumvar = 3;

/*
 * Function: MyFunction1
 * ----------------------------
 * Adding two number and dumpvalue and returns it
 *
 * a: Integer as first input
 * b: Integer as second input
 *
 * returns: a + b and also adding global dumvar
 *
 */
int MyFunction1(int a, int b)
{
    return (a + b + dumvar);
}

/*
 * Function: MyFunction2
 * ----------------------------
 * Checking if input is 0 or not and print result
 *
 * x: Integer as first input
 *
 * prints: print if x is null or not
 *
 */
void MyFunction2(float x)
{
    if(x == 0) cout<<"x is null"<<endl;
    else cout<<"x is not null"<<endl;
}
```

2 Exercises

2.1 Input and output in the console

In this question I had problems to use getline(cin...). I had to clear and flush the cin buffer.

Listing 3: Input and Output to console

```

/*
 * Function: ExampleInputOutput
 * ----------------------------
 * Showing usage of coun, cin and endl
 * Asks name and frist getting it with cin
 * Later asks name again and get that
 * with getline() function/
 * *** However I had problem with using getline,
 * maybe because of XCode, I had to use
 * cin.clear() to clean my cin buff before
 * getting the input, otherwise it was
 * skipping the input.
 *
 * prints: the input as name
 *
 */
void ExampleInputOutput()
{
    string input;
    cout<<"Please enter your name"<<endl;
    cin>>input;
    cout<<"Your name is: "<<input<<endl;

    input.empty();

    cout<<"We are sorry, we lost your name, can you re enter ?"<<endl;

    cin.clear();
    cin.ignore(INT_MAX, '\n');
    getline(cin, input);

    cout<<"Thank you, we got your name as: "<<input<<endl;
}
```

2.2 How to pass parameters to a function

2.2.1 On passing parameters by value or by reference

In this and following question, I used XOR swapping.

Listing 4: Swap Function by value and reference

```
/*
 * Function: swap_1
 * --------------------------------
 * Swapping the two integer inputs passed by value
 *
 * a: Interger number as first input
 * b: Integer number as second input
 *
 * returns: nothing.
 * prints: Prints the value of a and b before and after the swap
 *
 */
void swap_1(int a, int b)
{
    if(a == b) return;
    cout<<"Inside swap_1, a: "<<a<<" b: "<<b<<endl;
    a = a ^ b;
    b = a ^ b;
    a = a ^ b;
    cout<<"Finishing swap_1, a: "<<a<<" b: "<<b<<endl;
}

/*
 * Function: swap_2
 * --------------------------------
 * Swapping the two integer inputs passed by reference
 *
 * a: Interger number as first input
 * b: Integer number as second input
 *
 * returns: nothing.
 * prints: Prints the value of a and b before and after the swap
 *
 */
void swap_2(int& a, int& b)
{
    if(a == b) return;
    cout<<"Inside swap_2, a: "<<a<<" b: "<<b<<endl;
    a = a ^ b;
    b = b ^ a;
    a = a ^ b;
    cout<<"Finishing swap_2, a: "<<a<<" b: "<<b<<endl;
}
```

2.2.2 On passing parameters using pointers

Here is the similar swap function as above.

Listing 5: Swap Function using pointers

```
/*
 * Function: swap_1
 * ---------------------------------
 * Swapping the two integer inputs passed by pointer
 *
 * a: Pointer to integer number as first input
 * b: Pointer to integer number as second input
 *
 * returns: nothing.
 * prints: Prints the value of a and b before and after the swap
 *
 */
void swap_1(int *a, int *b)
{
    // If they equals we should/have to return, otherwise XOR method fails and makes both
        0
    if(*a == *b) return;
    cout<<"Inside swap_1, a: "<<*a<<" b: "<<*b<<endl;
    *a = *a ^ *b;
    *b = *b ^ *a;
    *a = *a ^ *b;
    cout<<"Finishing swap_1, a: "<<*a<<" b: "<<*b<<endl;
}
```

2.3 Multiple returned values

Here is the my function for 2 incoming input with pointers and 2 output parameters used to simulate returning multiple value.

Listing 6: Multiple Return Values

```
/*
 * Function: CartesianToPolar
 * ---------------------------------
 * Mapping complex number represented by z = a + bi to
 * polar system with P and theta components.
 *
 * a: Pointer to constant double number, first parameter, representing first part of the
     complex number
 * b: Pointer to constant double number, second parameter, representing second part of
     the complex number
 * p: Pointer to double number, Magnitude/Modus of the complex number
 * theta: Pointer to double number, angle of the complex number in polar form
 *
 * returns: nothing.
 * prints: nothing.
 *
 */
void CartesianToPolar(const double *a, const double *b, double *p, double *theta)
{
    *p = sqrt(pow(*a, 2.0) + pow(*b, 2.0));

    *theta = atan(*b / *a);

    return;
}
```

2.4 Default parameters

Here is the first declearation of the 2 function, and "IsMultipleOf" is used with default parameter for "2". Output is

```
100 is multiple of 2
55 is NOT multiple of 2
33 is multiple of 3
98 is NOT multiple of 3
```

Listing 7: Default Parameters

```cpp

bool IsMultipleOf(int, int = 2);
void checkMultiple(void);

/*
 * Function: IsMultipleOf
 * ----------------------------------
 * Checking if number p is a multiple of number q
 *
 * p: Integer number as first input p
 * q: Integer number as second input q | Default value if 2
 *
 * returns: boolean value(true if p is multiple of q, else false)
 * prints: nothing
 *
 */
bool IsMultipleOf(int p, int q)
{
    if(p % q == 0) return true;

    return false;
}

/*
 * Function: checkMultiple
 * ----------------------------------
 * Testing function for IsMultipleOf(int, int)
 *
 * returns: nothing.
 * prints: Call the "IsMultipleOf" and prints the result of multiplier check
 *
 */
void checkMultiple(void)
{
    if(IsMultipleOf(100))
        cout<<"100 is multiple of 2"<<endl;
    else
        cout<<"100 is NOT multiple of 2"<<endl;

    if(IsMultipleOf(55))
        cout<<"55 is multiple of 2"<<endl;
    else
        cout<<"55 is NOT multiple of 2"<<endl;

    // == //

    if(IsMultipleOf(33, 3))
        cout<<"33 is multiple of 3"<<endl;
    else
        cout<<"33 is NOT multiple of 3"<<endl;

    if(IsMultipleOf(98, 3))
        cout<<"98 is multiple of 3"<<endl;
    else
        cout<<"98 is NOT multiple of 3"<<endl;
}
```

2.5 Recursive functions

Here is recursive prime checker function, also using default parameter for second input, starts checking from 2 to prime number to check divisibility. Returns yes if the number is prime.

Listing 8: Recursive Prime Function

```

/*
 * Function: Prime
 * ----------------------------
 * Recursively finding if given number is prime
 *
 * prime: Integer number as first input for checking primeness
 * divider: Integer number as second input for recursively increasing to divide number
 *
 * returns: Boolean value, true if number is prime, otherwise false.
 * prints: nothing.
 *
 */
bool Prime(int prime, int divider)
{
    if(divider > prime) return false;
    if(prime % divider == 0 && prime > divider) return false;

    Prime(prime, divider + 1);

    return true;
}
```

2.6 Monodimensional array

Example that demonstration creating, filling and printing static and dynamic arrays.

Listing 9: Monodimensional Array Example

```

/*
 * Function: ArraysExample1
 * ----------------------------
 * Function for demonstrating monodimensional array example
 * Declearing static and dynamic array with size 10
 * Filling array with 0 to 9
 * Pringting values inside
 *
 * returns: nothing.
 * prints: nothing.
 *
 */
void ArraysExample1(void)
{
    int array1[10];
    int *array2 = new (nothrow) int[10];

    for (int i = 0; i < 10; i++) {
        array1[i] = i;
        array2[i] = i;
    }

    for (int i = 0; i < 10; i++) {
        cout<<"array1["<<i<<"] value is: "<<array1[i]<<endl;
        cout<<"array2["<<i<<"] value is: "<<array2[i]<<endl;
    }
}
```

2.7 Bidimensional array - Pascal's triangle revisited

In this example, I used both static and dynamic array same time. Tab2Static and Tab2Dynamic is the arrays, static one using [] operators however dynamic is only accessed with pointer aritmetic. This question is also printing Pascal's triangle with them.

Listing 10: Bidimensional Pascal Array

```

/*
 * Function: initArrays
 * ----------------------------
 * Initializing 2 array.
 * First assigns them to 0 then makes their first element 0
 *
 * arr1: 2 dimensional static integer array with size BI_ARRAY_SIZE * BI_ARRAY_SIZE
 * arr2: Pointer to 2 dimensional integer array with size BI_ARRAY_SIZE * BI_ARRAY_SIZE
 *
 * returns: nothing
 * prints: nothing.
 *
 */
void initArrays(int (*arr1)[BI_ARRAY_SIZE], int **arr2)
{
    // filling arr1 with 0 values according to its size
    memset(arr1, 0, sizeof(arr1[0][0]) * BI_ARRAY_SIZE * BI_ARRAY_SIZE);
    // As stated in the lab paper, we init first element with 1
    arr1[0][0] = 1;

    // initting the 2nd size of the arr2
    for (int i = 0; i < BI_ARRAY_SIZE; i++)
        *(arr2 + i) = new (nothrow) int[BI_ARRAY_SIZE];

    // As stated in lab paper, we init first element with 1
    *(*(arr2)) = 1;

    return;
}

/*
 * Function: fillArrays
 * ----------------------------
 * Fills 2 arrays with Pascal's triangle
 * Tab[i][0] = 1
 * Tab[i][j] = Tab[i - 1][j] + Tab[i - 1][j - 1] if i > 0, j > 0
 * Filling second array with pointer aritmetic.
 *
 * arr1: 2 dimensional static integer array with size BI_ARRAY_SIZE * BI_ARRAY_SIZE
 * arr2: Pointer to 2 dimensional interger array with size BI_ARRAY_SIZE * BI_ARRAY_SIZE
 *
 * returns: nothing
 * prints: nothing.
 *
 */
void fillArrays(int arr1[][BI_ARRAY_SIZE], int** arr2)
{
    // filling the arr1
    for (int i = 0; i < BI_ARRAY_SIZE; i++) {
        // As stated in formula in lab paper, we init [i][0] indexes to 1
        arr1[i][0] = 1;

        for (int j = 0; j < BI_ARRAY_SIZE; j++) {
            // If i or j equals to 0, we skip the formula
            if(!i || !j) continue;
            arr1[i][j] = arr1[i - 1][j] + arr1[i - 1][j - 1];
        }
    }

    // filling the arr2
    for (int i = 0; i < BI_ARRAY_SIZE; i++) {
        // As stated in formula in lab paper, we init [i][0] indexes to 1
        *(*(arr2 + i)) = 1;

        for (int j = 0; j < BI_ARRAY_SIZE; j++) {
            // If i or j equals to 0, we skip the formula
            if(!i || !j) continue;
            *(*(arr2 + i) + j) = *(*(arr2 + (i - 1)) + j) + *(*(arr2 + (i - 1)) + (j -1))
                ;
        }
    }


    return;
}

/*
 * Function: showBidimensionalArray
 * -----------------------------
 * Initializing, filling, and showing 2 dimensional 2 arrays.
 * Second array-Tab2Dynamic- is only used with pointer aritmetics no []
 *
 * returns: nothing
 * prints: nothing.
 *
 */
void showBidimensionalArray(void)
{
    // Our static array  decleration with given sizes
    int Tab2Static[BI_ARRAY_SIZE][BI_ARRAY_SIZE];
    // Our dynamic array decleration with given sizes
    int **Tab2Dynamic = new (nothrow) int*[BI_ARRAY_SIZE];;

    // Call for initing arrays
    initArrays(Tab2Static, Tab2Dynamic);
    // Call for filling the arrays and calculating the forumula
    fillArrays(Tab2Static, Tab2Dynamic);

    // Pringting the static array with index usage
    cout<<"Static array with index usage:"<<endl;
    for (int i = 0; i < BI_ARRAY_SIZE; i++) {
        for (int j = 0; j < BI_ARRAY_SIZE; j++) {
            cout<<Tab2Static[i][j]<<" ";
        }
        cout<<endl;
    }

    // Pringting the dynamic array with pointer aritmethic usage
    cout<<endl<<"-----------------"<<endl<<endl;
    cout<<"Dynamic array with pointer aritmethic usage:"<<endl;
    for (int i = 0; i < BI_ARRAY_SIZE; i++) {
        for (int j = 0; j < BI_ARRAY_SIZE; j++) {
            cout<<*(*(Tab2Dynamic + i) + j)<<" ";
        }
        cout<<endl;
    }
}
```

2.8 Multidimensional arrays as functions parameters(read and write)

As will be seen in the below, 3 matrix is defined in this examples with multidimensional arrays. They sent to "MultMatrix" function as array parameters, and C is filled with solution.

Listing 11: Multidimensional Array as function parameters

```

main() example:
    /*
     *      A       x       B       =       C
     *
     * | 1 0 0 |     | 1 2 3 |     | 1 2 3 |
     * | 0 1 0 | X | 4 5 6 | = | 4 5 6 |
     * | 0 0 1 |     | 7 8 9 |     | 7 8 9 |
     *
     */
    // Matrix A
    int A[3][3] = {
                    {1, 0, 0},
                    {0, 1, 0},
                    {0, 0, 1}};

    // Matrix B
    int B[3][3] = {
                    {1, 2, 3},
                    {4, 5, 6},
                    {7, 8, 9}};

    // Matrix C
    int C[3][3] = {0};


    MultMatrix(A, B, C);

 Here is the functions:

/*
 * Function: MultMatrix
 * ----------------------------
 * Given matrix A and B, calculates multiplication of them and assigns result to C
 *
 * A: 2 dimensional integer array with 3*3 size, representing matrix A
 * B: 2 dimensional integer array with 3*3 size, representing matrix B
 * C: 2 dimensional integer array with 3*3 size, representing result matrix C
 *
 * returns: nothing
 * prints: nothing.
 *
 * Method Used:
 * http://en.wikipedia.org/wiki/Loop_tiling
 * Original matrix multiplication:
 *   DO I = 1, M
 *     DO K = 1, M
 *       DO J = 1, M
 *         Z(J, I) = Z(J, I) + X(K, I) * Y(J, K)
 *
 */
void MultMatrix(int A[][3], int B[][3], int C[][3])
{
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 3; k++) {
                C[j][i] = C[j][i] + A[j][k] * B[k][i];
            }
        }
    }
}
```