University of Burgundy
Software Engineering
Tutorial No 2

SHAH Bhargav
DUDHAGARA Akshay

14 October 2018

1 Preliminaries

Listing 1: function.h file

```
#ifndef FUN1_H
#define FUN1_H

int ExampleInputOutput(int a, int b) // create a function
{
    int r = a + b;

    return (r);

}

#endif // FUN1_H
```

2 Exercices

2.1 Input and output in the console

Declare and Implement the function in .cpp file using .h file

Listing 2: ExampleInputOutput.cpp

```
#include <iostream>
//#include <string>
#include "fun1.h"
using namespace std;
int main()
{
    /*
     * call the function and use of cin,cout and endl
     */


    int a,b,z;
    cout<< "welcome in Lab2 :" << endl; // Cout is for print a statement and endl is for
        newline
```

```
14      cin >> a; // cin is the take the variable value
15      cin >> b;
16
17
18      z = ExampleInputOutput(a,b); //call function
19      cout << "The Result Is : " << z << endl;
20
21  }
```

2.2 How to pass parameters to a function

2.2.1 on passing parameters by value, Reference and Pointer

Listing 3: value/reference/pointer

```
1  #ifndef FUN2_H
2  #define FUN2_H
3
4  void swap_1(int a, int b) //swap function by value
5  {
6      int z;
7      z = a;
8      a = b;
9      b = z;
10
11 }
12 void swap_2(int &a, int&b) //swap function by Reference
13 {
14     int z;
15     z = a;
16     a = b;
17     b = z;
18 }
19 void swap_pointer(int* a, int* b) //swap function by pointer
20 {
21     int z;
22     z = *a;
23     *a = *b;
24     *b = z;
25 }
26 #endif // FUN2_H
27
28 #include<iostream>
29 #include "fun2.h"
30 using namespace std;
31
32 int main()
33 {
34     int i,j;
35
36     cin >> i >> j ;
37     swap_1(i,j);
38     cout << "swap_1 value: " << i << " " << j << endl;
39     swap_2(i,j);
40     cout << "swap_2 reference: " << i << " " << j << endl;
41     swap_pointer(&i, &j);
42     cout << "swap_pointer: "<< i << " " << j << endl;
43     return 0;
44 }
```

2.3 Multiple returned values

Listing 4: MultipleReturnedValues

```
#include<iostream>
#include<math.h> //declare a set of function to compute common mathematical operations
    and transformations
#include<stdio.h>
using namespace std;
/*
 *SHAH Bhargav & DUDHAGARA Akshay
 *  complex num eq = z = a + ib (a, b)
 * a = first parameter of complax number
 * b = second parameter of complex number
 * p = magnitude of complex number
 * theta = angle of complex number in polar form
 */
double & CartesianToPolar(double a, double b)
{
    double p = sqrt(a*a + b*b); // modulus
    double theta = atan(b/a);   //inbuid function compute arc tangent with two parameters

    return (p, theta); // return values

}

int main()
{
     double a, b, c, d;
     cout << "Enter value of a and b: " << endl;
     cin >> a >> b;
     (c, d) = CartesianToPolar(a, b);
     cout << "value of modulus and arctan is: " << c << " " << d << endl;
     return 0;


}
```

2.4 Defalut Parameters

Listing 5: Default parameters

```
#include <QCoreApplication>
#include <iostream>
/*
 * SHAH Bhargav & DUDHAGARA AkshayKumar
 *
 */
using namespace std;

int ismultipleof(int p, int q = 2) //create a function
{
    cout <<"value of p is: "<< p << " and value of q is " << q << endl;
    if(p % q == 0)
    {
        cout << "p divides q" << endl;
        return 0;
    }
    else
    {
        cout << "p does not divide q" << endl;
        return 0;
    }
}

int main()
{
    //implementing Ismultipleof()
    int p, q;
```

```
28      cout << "Enter the value of p and q to determines if a number p is a multiple of
            number q: " << endl;
29      cin >> p >> q;
30
31      ismultipleof(p);
32
33      cout << "Simulating value with value of q" << endl;
34      ismultipleof(p,q);
35  }
```

2.5 Recursive Functions

Listing 6: Recursive Functions

```
1  #include <iostream>
2  /*
3   * SHAH Bhargav & DUDHAGARA AkshayKumar
4   * num = input a number
5   *
6   */
7
8  using namespace std;
9
10 int isPrime(int num,int i) // create a function
11 {
12
13      if(i==1)
14      {
15          return 1;
16      }
17      else
18      {
19         if(num%i==0)
20           return 0;
21         else
22           isPrime(num,i-1);
23      }
24 }
25
26 int isprime1(int num, int i=2) // i value is fix
27
28 {
29      //cout << num << " " << i<< endl;
30      if(num%i == 0) // if number is not prime
31      {
32          cout << "Number is not prime" << endl;
33          return 0;
34      }
35      else
36      {
37          if((i>num/2))
38              return 1;
39          else
40              isprime1(num, i+1);
41      }
42 }
43
44 int main()
45 {
46 int num, prime;
47 cout << "Enter the value for which you need to find if it is prime number" << endl;
48 cin >> num;
49 prime = isprime1(num);
50
51 if(prime==1)
52      cout<< " is a prime number"<< num;
```

```
53 else
54     cout<< " is not a prime number"<<num;
55 }
```

2.6 Monodimensional Array

Listing 7: Monodimensional Array

```
1  #include<iostream>
2  using namespace std;
3  /*
4   * SHAH Bhargav & DUDHAGARA Akshay
5   * declare static and dynamic array with size 10 array with 0 to 9 and printing value
       inside
6   */
7  void ArrayExample1(void) //create a array function
8  {
9      int array1[10]; // number of element in array
10     int * array2 = new int[10];
11
12     for (int i = 0; i < 10; i++)
13     {
14         array1[i] = i;
15         array2[i] = i;
16
17     }
18     for (int i = 0; i < 10; i++)
19     {
20         cout << "array1["<< i << "] value is: "<< array1[i] << endl;
21         cout << "array2["<< i << "] value is: "<< array2[i] << endl;
22
23     }
24
25 }
26  int main()
27  {
28      ArrayExample1();
29      return 0;
30  }
```

2.7 Bidimensional array - Pascal's triangle revisited

Listing 8: Pascal Triangle revisited

```
1  #include <iostream>
2  using namespace std;
3  /*
4   * SHAH Bhargav & DUDHAGARA AkshayKumar
5   * Use a bidimensional array statically allocated
6   */
7
8  int main()
9  {
10     const int ROWS = 8, COLS = 8; // a static array with arbitary size
11     int Tab2[ROWS][COLS] ={} ; //bidimensional array Tab2 that contains n*n integers
12     Tab2[0][0] = 0;
13     cout << Tab2[0][0] << endl;
14
15
16     for ( int i = 1; i < ROWS; i++ )
17     {
18         Tab2[i][0] = 1; // when j = 0, set the values to 1
```

```
19          Tab2[i][i] = 1; // when j = i (end of triangle), set the values to 1
20          cout << Tab2[i][0] << "     ";
21
22          for ( int j = 1; j < COLS; j++ )
23          {
24              Tab2[i][j] = Tab2[i-1][j] + Tab2[i-1][j-1];
25              cout << Tab2[i][j] << "     ";
26          }
27
28          cout << Tab2[i][i] << endl; // display the result
29      }
30  }
```

2.8 Multidimensional array as functional parameters

Listing 9: Multidimensional array

```
1  #include <iostream>
2  using namespace std;
3  /*
4   * SHAH Bhargav & DUDHAGARA AkshayKumar
5   *
6   */
7  void  MultMatrix(int A[3][3],int B[3][3],int C[3][3]) // Create a statically allocated
       MultMatrix function
8  {
9      for(int i = 0; i < 3; i++)
10     {
11         for(int j = 0; j < 3; j++)
12         {
13             C[i][j] = 0;
14
15                 for(int k = 0; k < 3; k++)
16                 {
17                   C[i][j]+=A[i][k]*B[k][j];
18                 }
19         }
20     }
21 }
22
23
24 int main()
25
26 {
27     int A[3][3] = {{1,0,0},{0,1,0},{0,0,1}};  // Elements of Matrix A
28     int B[3][3] = {{1,2,3},{4,5,6},{7,8,9}}; // Elements of Matrix B
29     int C[3][3];
30
31 // For Matrix A
32     cout << "Matrix A is: " << endl;
33
34     for(int i = 0; i < 3; i++)
35     {
36         for(int j = 0; j < 3; j++)
37         {
38         cout << A[i][j] << " ";
39         }
40
41         cout << endl;
42     }
43
44 // For Matrix B
45     cout << "Matrix B is: " << endl;
46
47     for(int i = 0; i < 3; i++)
48     {
```

```
49          for(int j = 0; j < 3; j++)
50          {
51          cout << B[i][j] << " ";
52          }
53          cout << endl;
54      }
55
56  // Call MultMatrix function
57      MultMatrix(A,B,C);
58
59      //Final output in Matrix C
60      cout << "Matrix C is: " << endl;
61
62      for(int i = 0; i < 3; i++)
63      {
64          for(int j = 0; j < 3; j++)
65          {
66          cout << C[i][j] << " ";
67          }
68      cout << endl;
69      }
70
71      return 0;
72
73  }
```