University of Burgundy
Software Engineering
Tutorial No 3

SHAH Bhargav
DUDHAGARA AkshayKumar

28 October 2018

Exercise 1

Pointer As arrays

Listing 1: Pointer as arrays

```
1  // Exercise 1: Pointers as Arrays
2  // Declare two integer pointers a and b to dynamically allocate arrays of integers for n
       elements
3  void DisplayPointerInfo(int *a, int *b, int n)
4  {
5
6
7       // If Pointer is located at 'a' then value at each index is even numbers
8      cout << "The address of a Pointer is" << a << endl;
9      for(int i = 0; i < n; i++, a++)
10     {
11         cout << "Value at Index " << i << " is " << *a << endl;
12     }
13
14     /*
15         // Using []
16         for (int i = 0; i < n; i++)
17         {
18               cout << "Value at Index " << i << " is " << a[i] << endl;
19         }
20     */
21
22     // If Pointer is located at 'b' then value at each index is odd numbers
23     cout << "The address of a Pointer is " << b << endl;
24     for(int i = 0; i < n; i++, b++)
25     {
26         cout << "Value at Index " << i << " is " << *b << endl;
27     }
28     /*
29         // Using []
30         for (int i = 0; i < n; i++)
31         {
32               cout << "Value at Index " << i << " is " << b[i] << endl;
33         }
34     */
35 }
36 int main()
37 {
38
```

```
39      // Exercise 1
40      cout << "LAB 3" << endl;
41      cout << "Submitted by Bhargav SHAH and Akshaykumar DUDHAGARA" << endl;
42      cout << "Guided by Yohan Fougerolle " << endl;
43      cout << endl;
44      cout << endl;
45
46      cout << "Exercise 1" << endl;
47      int n;int *a;int *b;
48      cout << "Enter the number of elements" << endl;
49      cin >> n;
50
51      // Memory allocation
52      a = new int[n];
53      b = new int[n];
54
55      int a_index = 0, b_index = 0;
56
57      for(int i=0;i<2*n;i++)
58      {
59          if(i%2==0) // a will be filled with even number
60          {
61              a[a_index] = i;
62              a_index++;
63          }
64          else // b will be filled with odd number
65          {
66              b[b_index] = i;
67              b_index++;
68          }
69      }
70
71      // Display Info for pointer
72      DisplayPointerInfo(a,b,n);
```

Exercise 2

Swapping values of arrays represented by pointers

Listing 2: swapping values of arrays represented by pointer

```
1  // Exercise 2: Swapping values of arrays represented by pointers
2  // Declare a function swap1(...) that swaps two variables represented by pointers
3  void Swap1(int *c, int *d) // Two Variables c and d which are represented by pointers
4  {
5      int akkibhargav = *c;
6      *c = *d;
7      *d = akkibhargav;
8      /* *c = *c + *d;
9         *d = *c - *d;
10        *c = *c - *d; */
11 }
12
13 // Declare a function SwapArray(...) that swaps all the values of two arrays represented
       by pointers
14 void SwapArray(int *a, int *b, int n)
15 {
16     for(int i = 0; i < n; i++)
17     {
18         int akkibhargav = a[i];
19         a[i] = b[i];
20         b[i] = akkibhargav;
21         /* a[i] = a[i] + b[i];
22            b[i] = a[i] - b[i];
23            a[i] = a[i] - b[i]; */
```

```
24
25
26            // without using []
27             /*
28                  for( int i = 0; i < n; i++, a++, b++)
29                       {
30                          Swap1(a,b);
31                       }
32              */
33      }
34  }
35  // Exercise 2 {Int main() file}
36
37      cout << endl;
38      cout << "Exercise 2" << endl;
39      cout << "Swaps two variables represented by pointers" << endl;
40
41      // Randomly take c1=5 and d1=7
42      int c1(5),d1(7);
43      int *c = &c1, *d = &d1;
44
45      // Before Swapping
46      cout << "Before Swapping:" << " c=" << *c << "," << " d=" << *d << endl;
47
48      // Swaps two variables represented by pointers
49      Swap1(c,d);
50
51      // After Swapping
52      cout << "After Swapping:" << " c=" << *c << "," << " d=" << *d << endl;
53
54      // Swaps all the values of two arrays represented by pointers
55      cout <<endl;
56      cout << "Swaps all the values of two arrays represented by pointers" << endl;
57      cout << "Before Swapping:" << endl;
58
59      // Use exe.1,Display Info for pointer
60      DisplayPointerInfo(a,b,n);
61
62      SwapArray( a, b, n);
63
64      cout << endl;
65      // After Swapping
66      cout << "After Swapping:" << endl;
67
68      // If Pointer is located at 'a'
69      cout << "Pointer is located at " << a << endl;
70      for(int i(0); i<n ; i++)
71      {
72          cout << "Value at Index " << i << " is " << a[i] << endl;
73      }
74
75      // If Pointer is located at 'b'
76      cout << "Pointer is located at " << b << endl;
77      for(int i=0; i<n;i++)
78      {
79          cout << "Value at index " << i << " is " << b[i] << endl;
80      }
```

Exercise 3

Allocation and deallocation of mono-dimensional and bi-dimensional arrays represented by pointers

Listing 3: arrays represented by pointers

```
1  // Exercise 3: Allocation and deallocation of monodimensional and bidimensional arrays
       represented by pointers
2  // 1.Declare a function CreateArray(...) that returns a pointer to an array of n integers
3  int * CreateArray(int n) // Create array
4  {
5      //static int a[] = {1,2,3,4,5}; return a;
6
7      return (new int [n]);
8  }
9
10 void deleteArray(int*a[]) // Delete array
11 {
12     delete [] a;
13 }
14
15 // 2.Declare a function CreateMatrix(...) that returns a pointer to an array of arrays of
       n*m floats
16 float ** CreateMatrix(int n, int m) // create matrix
17 {
18     float**array = new float*[n];
19     for(int i = 0; i < n; i++)
20         array[i] = new float[m];
21 //     for(int i = 0; i < m; i++)
22 //         for(int j = 0; j < n; j++)
23 //           array[i][j] = ((i*n)+j);
24     return array;
25 }
26
27 void deleteMatrix(float **array, int n, int m) // detete matrix
28 {
29     for(int i = 0; i < n; i++)
30         delete [] array[i];
31     delete [] array;
32 }
33
34 // 3.Declare a function DisplayMatrix(...) that displays the address of the matrix of
       floats
35 void DisplayMatrix(float **av, int n, int m) // display matrix
36 {
37     cout << "Address of matrix of floats is" << av << endl;
38     for(int i = 0; i < n; i++)
39     {
40         for(int j = 0; j < m; j++)
41             cout<< av[i][j] << " ";
42         cout << endl;
43     }
44 }
45
46  // Exercise 3 {int main() file}
47
48     cout << endl;
49     cout << "Exercise 3" << endl;
50     cout << "1. Pointer to an array of n integers" << endl;
51
52     int * ptr;
53     ptr = CreateArray(n);
54     DisplayPointerInfo(ptr,ptr,n);
55
56     //deleteArray(ptr); // could not print because array is deleted
57
58     cout << endl;
```

```
59      cout << "2. Pointer to an array of arrays of n*m floats" << endl;
60      float ** p = CreateMatrix(n,n); // To create matrix
61
62      //deleteMatrix(p,n,m);
63
64      DisplayMatrix(p,n,n); // To display matrix
```

Exercise 4

A little bit of geometry

Listing 4: geometry

```
1  // Exercise 4: A little bit of geometry
2  // Declare functions to compute the dot product
3  float dot_product ( float *a, float *b, int n)
4  {
5      float product(0);
6      for (int i = 0; i < n; i++)
7          product += a[i]*b[i]; // dot product of a and b is a*b
8
9      return product;
10
11     // without using []
12     /*
13       for (int i = 0; i < n; i++, a++, b++)
14           product += (*A)*(*B);
15     */
16 }
17
18 // Declare functions to compute the inner product
19 float *inner_product(float *a, float *b, int n)
20 {
21     if ( n!= 3)
22         return NULL;
23
24     float *product = new float[3];
25     // calculate inner product
26     product[0] =  a[1] * b[2] - a[2]*b[1];
27     product[1] =  a[2] * b[0] - a[0]*b[2];
28     product[2] =  a[0] * b[1] - a[1]*b[0];
29 }
30             // {int main() file start from here } //
31 cout << endl;
32     cout << "Exercice 4"<<endl;
33
34     float *P = new float[3];
35     float *Q = new float[3];
36
37
38     P[0] = 1; P[1] = 1; P[2] = 0; // P = (1,1,0)
39     Q[0] = 1; Q[1] = 1; Q[2] = 1; // Q = (1,1,1)
40
41     // Dot product
42     cout << "Dot Product is:" << endl;
43     cout<< dot_product(P,Q,3) << endl;
44
45     // Inner product
46     float *R = inner_product(P,Q,3);
47     for (int i=0; i<n; i++, R++)
48     {
49         cout << "value at index"<< i << " is "<< *R <<endl;
50     }
```

Exercise 5

Matrix multiplication in general case

Listing 5: Matrix Multiplication

```
// Exercise 5: Matrix multiplication in the general case
// Declare a function MatrixProduct(...) that returns the matricial product of two
    matrices of arbitraty dimensions
void MatrixProduct(int r1, int r2, int col1, int c2, int D[][5], int E[][5])
{
    int i, j, k;
    int C[r1][c2];
// Using for loop multiplying two matrices
    for (i = 0; i < r1; i++)
    {
        for (j = 0; j < c2; j++)
        {
            C[i][j] = 0;
            for (k = 0; k < r2; k++)
            {
                *(*(C + i) + j) += *(*(D + i) + k) *
                                   *(*(E + k) + j);
                //C[i][j] += D[i][k] * E[k][j];
            }
        }
    }

    for (i = 0; i < r1; i++)
    {
        for (j = 0; j < c2; j++)
        {
            cout << *(*(C + i) + j) << " ";
        }
        cout << "\n";
    }

}
// Exercise 5  (int main() file start from here)

    cout << endl;
    cout << "Exercice 5" << endl;

    int r1,col1,r2, c2,i, j;
    int D[5][5], E[5][5];

    // Create Matrix A
    cout << "Enter number of rows and columns of matrix A : ";
    cin >> r1 >> col1;

    // Create Matrix B
    cout << "Enter number of rows and columns of matrix B : ";
    cin >> r2 >> c2;

    if (col1 != r2)
    {
        cout << "Matrices cannot be multiplied!";
        exit(0);
    }

    // Storing element of Matrix A
    cout << "Enter elements of matrix A : ";
    for (i = 0; i < r1; i++)
       for (j = 0; j < col1; j++)
           cin >> D[i][j];

    // Storing element of Matrix B
    cout << "Enter elements of matrix B : ";
```

```
62      for (i = 0; i < r2; i++)
63          for (j = 0; j < c2; j++)
64              cin >> E[i][j];
65
66      // Product of Two matrices
67      cout << "Product of matrices\n";
68      MatrixProduct(r1, r2, col1, c2, D, E);
69
70
71
72      return 0;
73
74  }
```