University of Burgundy

Software Engineering

Tutorial 3

Lab Report-3

Author:
PARMAR Hardiksinh

Supervisor:
Dr. Yohan Fougerolle

October 28, 2018

1 Pointer as arrays

1.1 Declare and implement func DisplayPointerInfo()

```
#include <iostream>
#include "lab_3.h"

using namespace std;
//1.a> display pointers as varaibles----------------------

void DisplayPointerInfo (int *pntr, int length)
{
    int i;
    for (i = 0; i < length; i++)
    {
        cout << *pntr++ << " ";
    }
    cout << endl;
}
```

1.2 Dynamically allocate n integers:: odd and even

```
// 1.b> pointers dynamic array to find odd and even----------

    int n;
    int *c, *d;

    cout << "Enter no. of values for the array here: " << endl;
    cin >> n;
    c = new int [n]                 // dynamic arrays
    d = new int [n];
                        //reading arrays by pointers
        for (int i = 0; i < n ; i++)
        {
            *(c + i) = i * 2;
            *(d + i) = i * 2 + 1;
        }
```

2 Swapping arrays by Pointers

2.1 Swapping variables by Pointers

```
//2.a> swapping values of values in a variable by pointers -----
void swap (int *xx, int *yy)
```

```
{
    int *t = xx;
    xx = yy;
    yy = t;
}
```

2.2 Swapping arrays by pointers

```
//2.b> pointers as arrays :: SwapArray————————————————————

void SwapArray (int *xx, int *yy, int length)
{
    int temp, i;
    for (i = 0; i < length; i++)
        {
          temp = *xx;
          *xx = *yy;
          *yy = temp;
          *xx++;
          *yy++;
        }
}
```

3 Allocation and Deallocation of 1D, 2D arrays by pointers

3.1 CreateArray() returns pointer of array

```
 //3.1 Creating 1D array of n integers by pointers ——————————

 int* CreateArray (int n)
{
   int *pntr = new int[n];
   return pntr;
}
```

3.2 DeleteArray() of n integers

```
 //3.2 Deleting 1D array of n integers by pointers————————————

  void DeleteArray(int* p)
  {
      delete p;
  }
```

3.3 CreateMatrix() of n X m floats

```
//3.3 Creating a matrix n X m integers by pointers————————

int** CreateMatrix(int n, int m)
{
    int **a;
    a = new int* [n];
    for(int i = 0; i < n; ++i)
    a[i] = new int [m];
    return a;
}
```

3.4 DeleteMatrix() of n X m floats

```
//3.4 Deleting a matrix by pointers ————————————

 void DeleteMatrix(int** p)
 {
     delete p;
 }
```

3.5 DisplayMatrix() of n X m floats with address

```
//3.5 Dispplaying a matrix n X m integers by pointers———

 void DisplayMatrix(int** p, int q, int r)
{
  for (int i = 0; i < q; ++i)
  {
      for(int j = 0; j < r; ++j)
      {
          cout << *(*(p + i) + j) << "␣";
      }
      cout << endl;
  }
}
```

4 A little bit of Geometry

4.1 Finding Dotproduct and inner product in 3D

```
//Finding Dot product and innerproduct of 3D
  float dot_product ( float *a, float *b, int n)
```

```
{
  float product(0);
  for (int i = 0; i < n; i++)
      product += a[i]*b[i];
      return product;
 }

 float *inner_product(float *a, float *b, int n)
{
  if ( n!= 3) return NULL;
      float *product = new float[3];
          product[0] = a[1] * b[2] - a[2]*b[1];
          product[1] = a[2] * b[0] - a[0]*b[2];
          product[2] = a[0] * b[1] - a[1]*b[0];
}
```

4.2 main function for all the above programs

```
//main function for all the above programs

#include <iostream>
#include "lab_3.h"
using namespace std;

int main()
{
   int a = 66, b = 34;
   int *pa = &a, *pb = &b;           // inputs to array


  //pointers as varaibles----------------------
 //values before swapping
 cout << "Value a before swap: " << a << endl;
 cout << "Value b before swap: " << b << endl;
   swap(*pa, *pb);
 //values after swapping
 cout << "Value a after swap: " << a << endl;
 cout << "Value b after swap: " << b << endl;

   //1.a> pointers as arrays ------------------------

   int arrA[] = {11, 22, 33, 44, 55};
   int arrB[] = {13, 15, 17, 19, 21};
```

```
// 1.b> pointers dynamic array to find odd and even numbers------

    int n;
    int *c, *d;

    cout << "Enter no. of values for the array here: " << endl;
    cin >> n;
    c = new int [n];     // dynamic arrays
    d = new int [n];
                         //reading arrays by pointers
        for (int i = 0; i < n ; i++)
        {
            *(c + i) = i * 2;
            *(d + i) = i * 2 + 1;
        }

    cout << "The even array are: " << endl;//displaying pointers
    DisplayPointerInfo(c, n);
    cout << "The odd array are: " << endl;
    DisplayPointerInfo(d, n);

    cout<<"  "<< endl;

   //2. swapping values of arrays by pointers ----------------

  //values in arrays before swapping

  cout << "Array A before swap: " << endl;
  DisplayPointerInfo(arrA, 5);

  cout << "Array B before swap: " << endl;
  DisplayPointerInfo(arrB, 5);

  SwapArray(arrA,arrB, 5);
  cout<<"  "<< endl;

  //values in arrays after swapping

  cout << "Array A after swap: " << endl;
  DisplayPointerInfo(arrA, 5);

  cout << "Array B after swap: " << endl;
  DisplayPointerInfo(arrB, 5);
  cout<<"  "<< endl;
```

```
//3.1 Creating 1D array of n integers by pointers ----------

int x;
cout << "length of the array is: " << endl;
cin >> x;
int *pntr1 = CreateArray(x); //creating an array
for (int i = 0 ; i < x ; i++)
    {
        *(pntr1+1)=i;
    }
DisplayPointerInfo(pntr1, x);

//3.2 Deleting 1D array of n integers by pointers ----

DeleteArray(pntr1);          //deleting an array


//3.3 Creating a matrix n X m integers by pointers-----

int **p, h, w;
    cout << "row of the matrix is: " << endl;
    cin >> h;
    cout << "column of the matrix is: " << endl;
    cin >> w;


    p = CreateMatrix(h, w);
    DisplayMatrix(p, h, w);

 //3.4 Deleting a matrix by pointers ----------

    DeleteMatrix(p);

   //Finding Dot product and innerproduct of 3D
    float *P = new float[3];
    float *Q = new float[3];
    P[0] = 1;
    P[1] = 1;
    P[2] = 0;
    Q[0] = 1;
    Q[1] = 1;
    Q[2] = 1;

    cout<< "Dot Product is: ----- " << endl;
    cout<< dot_product(P,Q,3) << endl;
```

```
        cout<< "Inner Product is:    " << endl;
         cout<< " " << endl;

        float *R = inner_product(P,Q,3);
        for (int i=0; i<n; i++, R++)
        {
            cout<< "value at index" << i << " is "<< *R << endl;
        }

}
```

5 Multiplication of matrices in general

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

void enterData(int firMat[][10], int secMat[][10], int rowFir,
        int colFir, int rowSec, int colSec);
void multiplyMatrices(int firtMat[][10], int secMat[][10], int
multRes[][10], int rowFir, int columnFir, int rowSec,
        int colSec);
void display(int mult[][10], int rowFir, int colSec);

//main() for matrix
int main()
{
    int firMat[10][10], secMat[10][10], mult[10][10], rowFirst,
          colFir, rowSec, colSec, i, j, k;

    cout << "Enter rows and column for first matrix: ";
    cin >> rowFirst >> colFir;

    cout << "Enter rows and column for second matrix: ";
    cin >> rowSec >> colSec;

    // If colum of first matrix in not equal to row of second
    matrix, asking user to enter the size of matrix again.
    while (colFir != rowSec)
    {
    cout << "Error! column of first matrix not equal to row of
              second." << endl;
        cout << "Enter rows and column for first matrix: ";
        cin >> rowFirst >> colFir;
        cout << "Enter rows and column for second matrix: ";
```

```
        cin >> rowSec >> colSec;
    }

    // Function to take matrices data
    enterData(firMat, secMat, rowFirst, colFir, rowSec, colSec);

    // Function to multiply two matrices.
    multiplyMatrices(firMat, secMat, mult, rowFirst, colFir,
    rowSec, colSec);

    // Function to display resultant matrix after
    //multiplication.
    display(mult, rowFirst, colSec);

    return 0;
}

void enterData(int firMat[][10], int secMat[][10], int rowFir,
    int colFir, int rowSec, int colSec)
{
    int i, j;
    cout << endl << "Enter elements of matrix 1:" << endl;
    for(i = 0; i < rowFir; ++i)
    {
        for(j = 0; j < colFir; ++j)
        {
            cout << "Enter elements a"<< i + 1 << j + 1 << ": ";
            cin >> firMat[i][j];
        }
    }

    cout << endl << "Enter elements of matrix 2:" << endl;
    for(i = 0; i < rowSec; ++i)
    {
        for(j = 0; j < colSec; ++j)
        {
            cout << "Enter elements b" << i + 1 << j + 1 << ": ";
            cin >> secMat[i][j];
        }
    }
}

void multiplyMatrices(int firMat[][10], int secMat[][10],
int mult[][10], int rowFir, int colFir, int rowSec, int colSec)
{
```

```
    int i, j, k;

    // Initializing elements of matrix mult to 0.
    for(i = 0; i < rowFir; ++i)
    {
        for(j = 0; j < colSec; ++j)
        {
            mult[i][j] = 0;
        }
    }

    // Multiplying matrix firstMatrix and secondMatrix and
    storing in array mult.

    for(i = 0; i < rowFir; ++i)
    {
        for(j = 0; j < colSec; ++j)
        {
            for(k=0; k<colFir; ++k)
            {
                mult[i][j] += firMat[i][k] * secMat[k][j];
            }
        }
    }
}

void display(int mult[][10], int rowFirst, int columnSecond)
{
    int i, j;

    cout << "Output Matrix:" << endl;
    for(i = 0; i < rowFirst; ++i)
    {
        for(j = 0; j < columnSecond; ++j)
        {
            cout << mult[i][j] << " ";
            if(j == columnSecond - 1)
                cout << endl << endl;
        }
    }
}
```

6 Outputs of all the programs

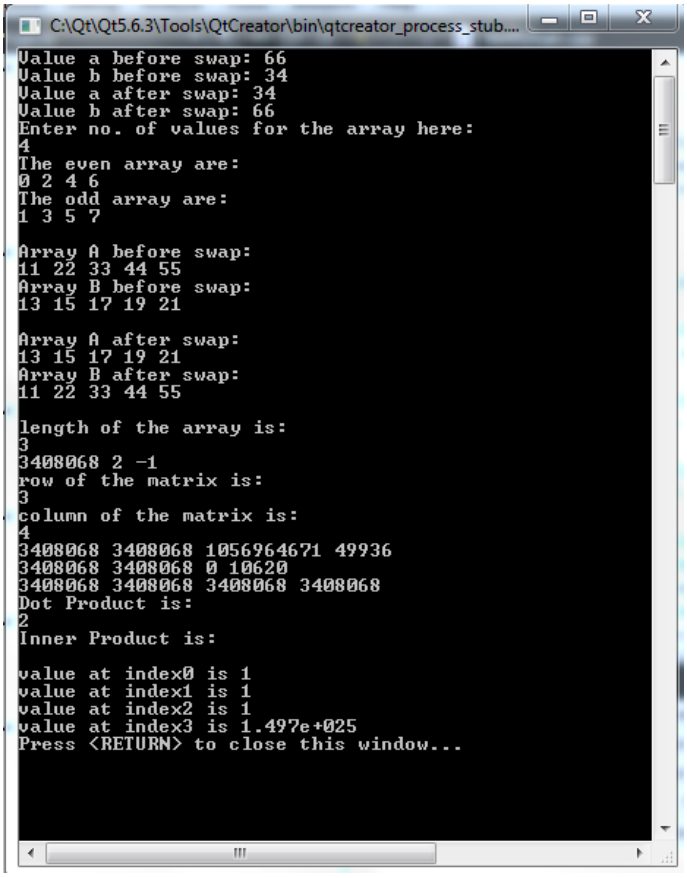

Figure 1: All outputs

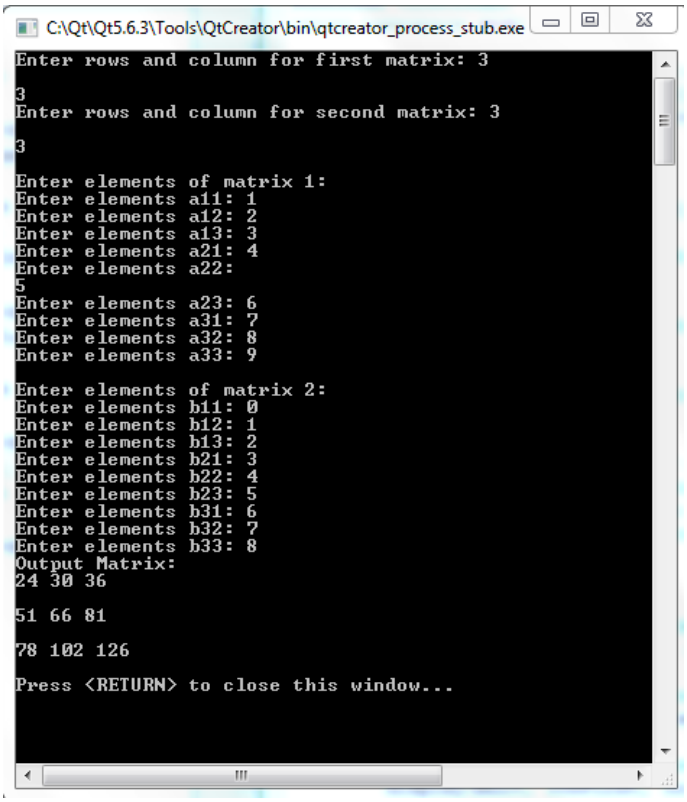

Figure 2: output of the MatMul Program