SSCM 1313

C++ COMPUTER PROGRAMMING

Chapter 5:

Structure and Array Applications

Authors:

Farhana Johar

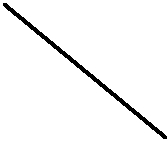
Professor Dr. Shaharuddin Salleh

Structure

* A structure groups variables into a hierarchical tree based on their common ancestral origin. Under this arrangement, a structure starts with a *parent* which includes several *children*.
* Constructed using tydef struct
* A structure must have a name
* Declared in the pre-processing area

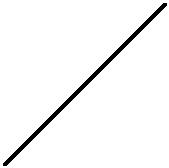
Example

|  |  |  |  |
| --- | --- | --- | --- |
| typedef struct |  |  |  |
| { |  |  |  |
| char name[10]; |  | *members* |  |
| int age; |  |  |
|  |  |  |
| double cpa; |  |  |  |
| } CAMPUS; |  |  |  |
|  |  |  |



*name of structure*

*object of the structure*



|  |  |  |  |
| --- | --- | --- | --- |
| CAMPUS student; |  |  |  |
| strcpy(student.name,"Michael"); | | *assignment of values* |  |
| student.age=35; | |  |
| *to members* |  |
| student.cpa=3.15; | |  |
|  |  |
|  |  |  |  |

Points in Cartesian Coordinates

typedef struct

{

double x,y; } POINT;

POINT pt[10];



Code5A.cpp:Structure representing a point.

#include <iostream> #define n 4

using namespace std;

void main()

{

int i,j;

typedef struct

{

double x,y;

} POINT;

POINT pt[n+1]; double Eu[n+1][n+1];

for (i=1;i<=n;i++)

{

pt[i].x=(double)2\*i;

pt[i].y=(double)3\*i‐1;

}

cout << "Euclidean distance between points i and j" << endl;

for (i=1;i<=n;i++)

for (j=1;j<=n;j++)

{

Eu[i][j]=sqrt(pow(pt[i].x‐pt[j].x,2)+pow(pt[i].y‐pt[j].y,2));

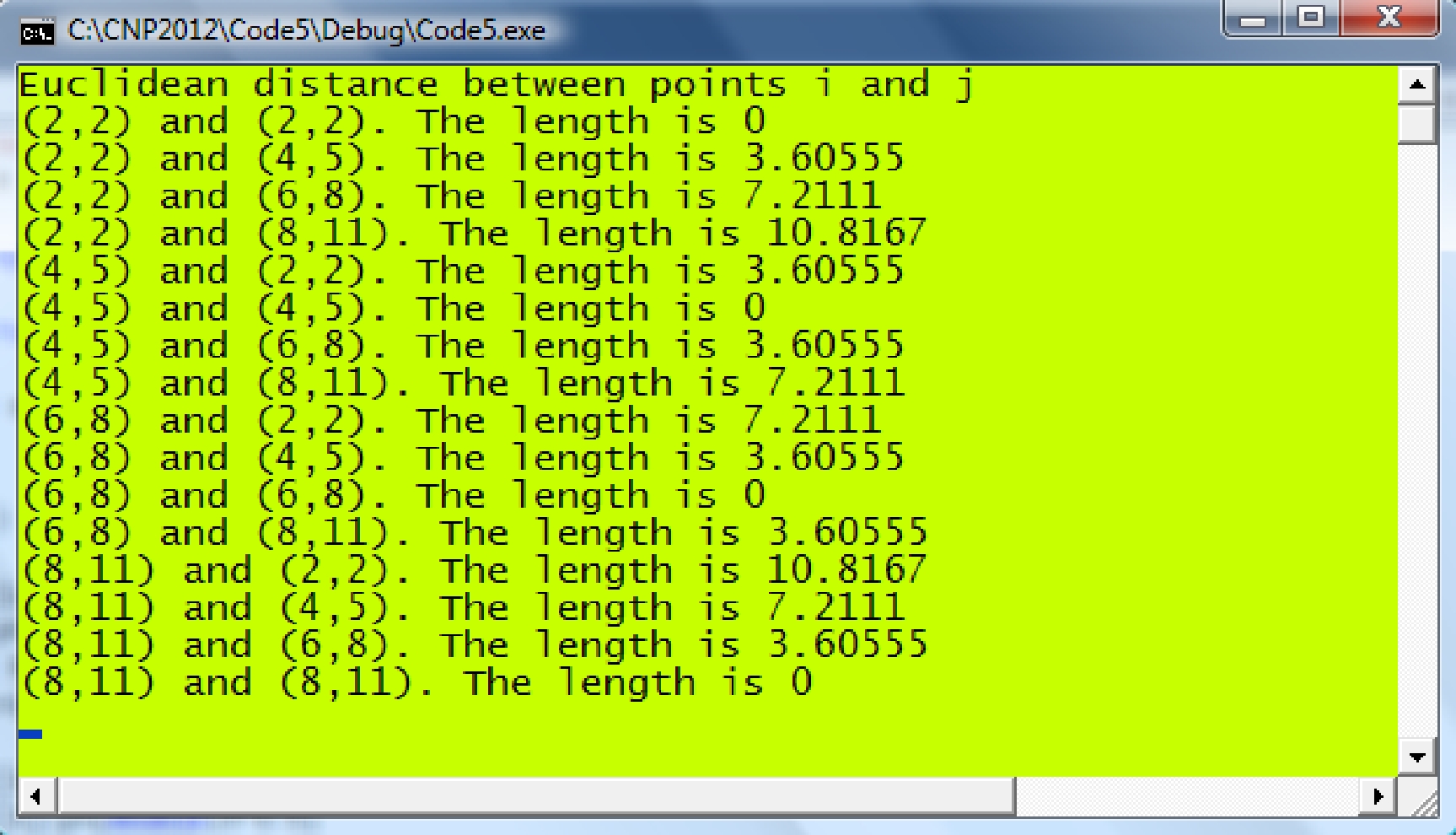
cout << "(" << pt[i].x << "," << pt[i].y << ") and ("<<pt[j].x

<<","<< pt[j].y <<").The length is "<<Eu[i][j] << endl;

}

cin.get();

}



Nested Structure

typedef struct

{

double x,y;

} POINT;

typedef struct

{

POINT b,e;

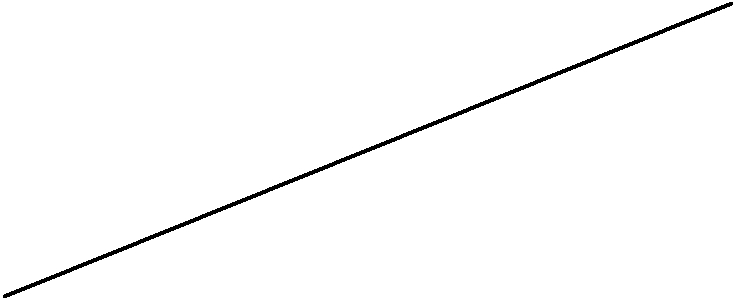
} LINE;

*Description:*

A line has two end points.

Each point in a line is represented as (x,y) coordinates.

LINE Ln;

Ln.b.x=3; Ln.b.y=‐1; e

Ln.e.x=7; Ln.e.y=0; (e.x,e.y)

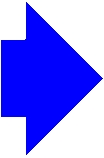
b

(b.x,b.y)

typedef struct

{

double x,y; } POINT;



typedef struct

{

POINT b,e; } LINE;

typedef struct

{

LINE p,q,r; } TRIANGLE;

*A triangle has three points.*

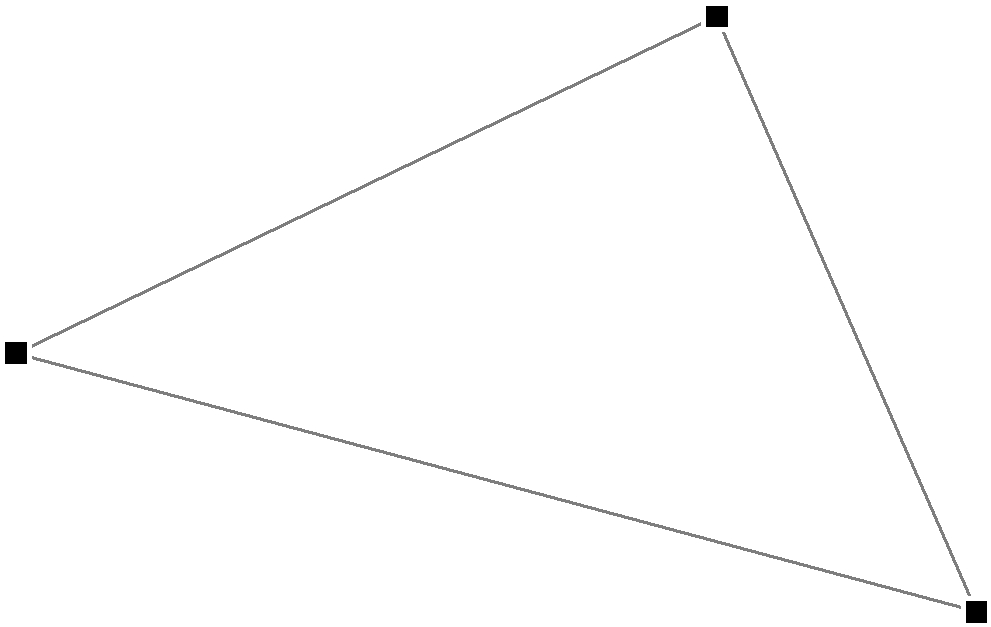
*Each line in a triangle has two end points.*

*Each point in a line is represented as (x,y) coordinates.*

t.p.b.x=3; t.p.b.y=‐1; t.p.e.x=7; t.p.e.y=0; t.q.b.x=7; t.q.b.y=0; t.q.e.x=5; t.q.e.y=‐8; t.r.b.x=5; t.r.b.y=‐8; t.r.e.x=3; t.r.e.y=‐1;

(5,-8)

t.r.b t.q.e



t.r

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| t.r.e |  | t | t.q |  |
|  |  |  |
| (3,-1) |  |  |  |  |
| t.p.b |  |  |  |  |
|  | t.p |  | t.q.b |  |
|  |  |  |  |
|  |  |  | (7,0) |  |
|  |  |  | t.p.e |  |

**Figure 5.1.** TRIANGLE and its nested structures.

Code5B.cpp: Structure representing a triangle.

#include <iostream> #include <iomanip> #include <time.h> #define n 3

using namespace std;

void main()

{

int i; srand(time(0)); typedef struct

{

double x,y; } POINT;

typedef struct

{

POINT b,e; double length;

} LINE;

typedef struct

{

LINE p,q,r; double area;

} TRIANGLE; TRIANGLE t[n+1];

for (i=1;i<=n;i++)

{

t[i].p.b.x=(double)(rand()%50); t[i].p.b.y=(double)(rand()%50); t[i].p.e.x=(double)(rand()%50); t[i].p.e.y=(double)(rand()%50); t[i].q.e.x=(double)(rand()%50); t[i].q.e.y=(double)(rand()%50); t[i].q.b.x=t[i].p.e.x; t[i].q.b.y=t[i].p.e.y; t[i].r.b.x=t[i].q.e.x; t[i].r.b.y=t[i].q.e.y; t[i].r.e.x=t[i].p.b.x; t[i].r.e.y=t[i].p.b.y; t[i].area=0.5\*(‐t[i].p.e.x\*t[i].p.b.y+t[i].q.e.x\*t[i].p.b.y

+t[i].p.b.x\*t[i].p.e.y‐t[i].q.e.x\*t[i].p.e.y ‐t[i].p.b.x\*t[i].q.e.y+ t[i].p.e.x\*t[i].q.e.y);

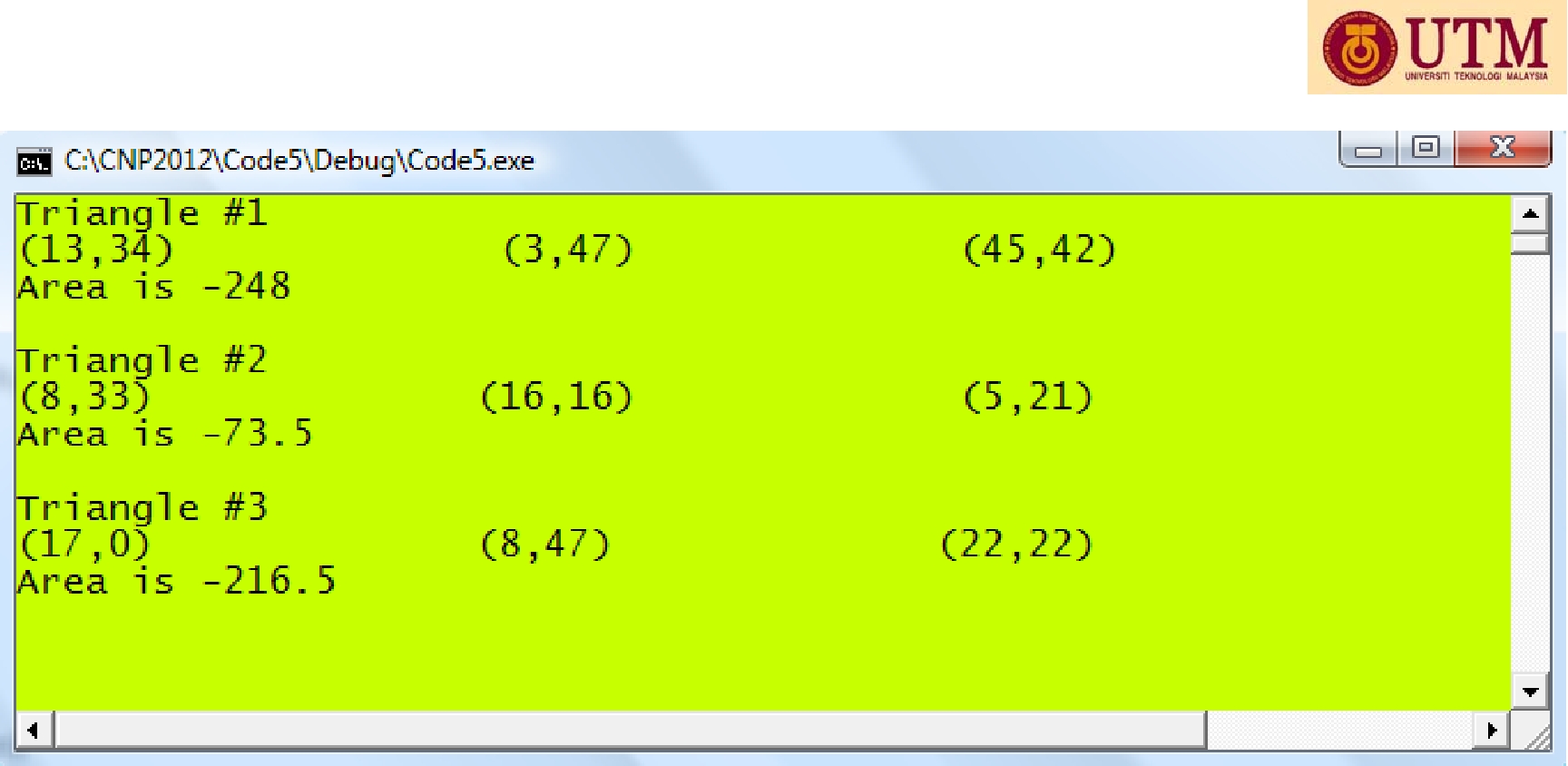
cout << "Triangle #" << i << endl;

cout << "(" << t[i].p.b.x << "," << t[i].p.b.y << ")" << setw(15); cout << "(" << t[i].q.b.x << "," << t[i].q.b.y << ")" << setw(15); cout << "(" << t[i].r.b.x << "," << t[i].r.b.y << ")" << endl; cout << "Area is " << t[i].area << endl << endl;

}

cin.get();

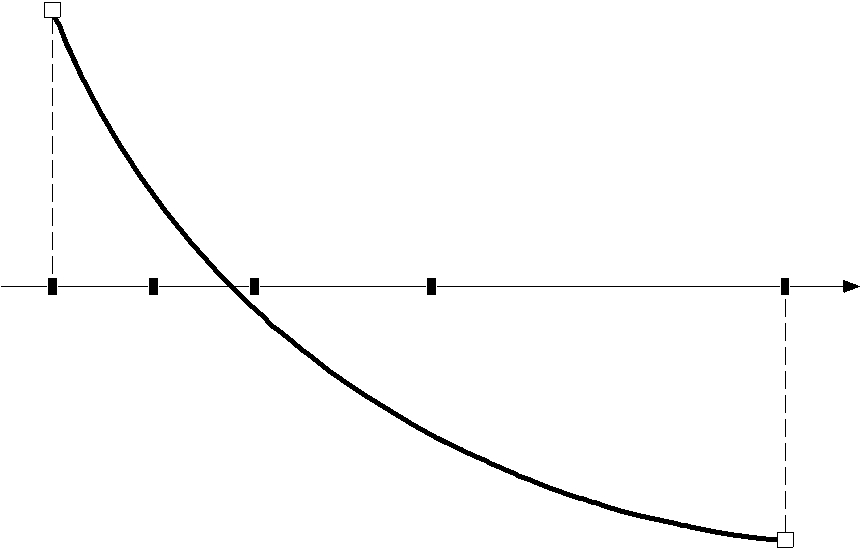
}



CS1 (Case Study 1): Finding the root of a function

*Given f* (*x*)0 *, find the value of x , assuming the value exists.*

*f* (*a*0)



|  |  |  |
| --- | --- | --- |
| *c*1 | *c*0 | *b*0 |

*x*

|  |  |
| --- | --- |
| *a*0 | *c*2 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | *f* (*b*0) | |  |
|  | *a*3 |  | *b*3 | | |  |
| *i =* 3 | |  |  |  |  |  |
|  |  |  |  |  |
| *a*2 | |  | *b*2 | | |  |
| *i =* 2 |  |  |  |  |  |  |
|  |  |  |  |  |  |
| *a*1 | |  |  | *b*1 | |  |
| *i =* 1 |  |  |  |  |  |  |
|  |  |  |  |  |  |
| *a*0 | |  |  |  | *b*0 |  |
| *i =* 0 |  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm 5.1.** Bisection Method. | | | | | |  |  |  |  |
| Given | *f* (*x*)0, *ε* | |  | and the initial end-points [*a*0 , *b*0 ] | | | where | *f* (*a*0) *f* (*b*0)0; |  |
| Given *max* =maximum number of iterations; | | | | | | |  |  |  |
| For *i*  0 to *max* | | | *a* *b* | |  |  |  |  |  |
| Compute *ci*  | | |  |  |  |  |  |
|  | *i* 2 *i* ; |  |  |  |  |  |
| If *f* *ai*  *f* *ci*  0 | | | | |  |  |  |  |  |
|  | Update *bi* 1  *ci* | | | | and *ai* 1  *ai* | ; |  |  |  |
| If *f* *ai*  *f* *ci*  0 | | | | |  |  |  |  |  |
|  | Update *ai* 1  *ci* | | | | and *bi* 1  *bi* | ; |  |  |  |
|  | *ci* − *ci* −1 |  ε | |  |  |  |  |  |  |
| If |  |  |  |  |  |  |
|  | Solution= *ci* | | | ; |  |  |  |  |  |
| Endfor | Stop the iterations; | | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Code5C.cpp: Finding the root of a function.

#include <iostream> #include <iomanip>

#define f(x) (pow(x,3)‐pow(x,2)‐2)

#define N 20

#define EPSILON 0.005

using namespace std;

void main()

{

double \*a, \*b, \*c; a=new double [N+1]; b=new double [N+1]; c=new double [N+1]; a[0]=1; b[0]=2;

cout<<"i"<<setw(15)<<"a"<<setw(15)<<"b"<<setw(15)<<"c"<<setw(15)<<"error"<<endl;

for (int i=0;i<=N;i++)

{

|  |  |
| --- | --- |
| c[i]=(a[i]+b[i])/2; |  |
| if (f(a[i])\*f(c[i])>0) |  |
| a[i+1]=c[i]; b[i+1]=b[i]; | // update a |
| else |  |
| b[i+1]=c[i]; a[i+1]=a[i]; | // update b |
| if (i>0) |  |
| { |  |

double StopError = fabs(c[i]‐c[i‐1]);

cout << i << setw(15)<< a[i]<<setw(15)<< setw(15)

<< b[i] << setw(15) << c[i] << setw(15) << StopError << endl;

if (StopError<EPSILON)

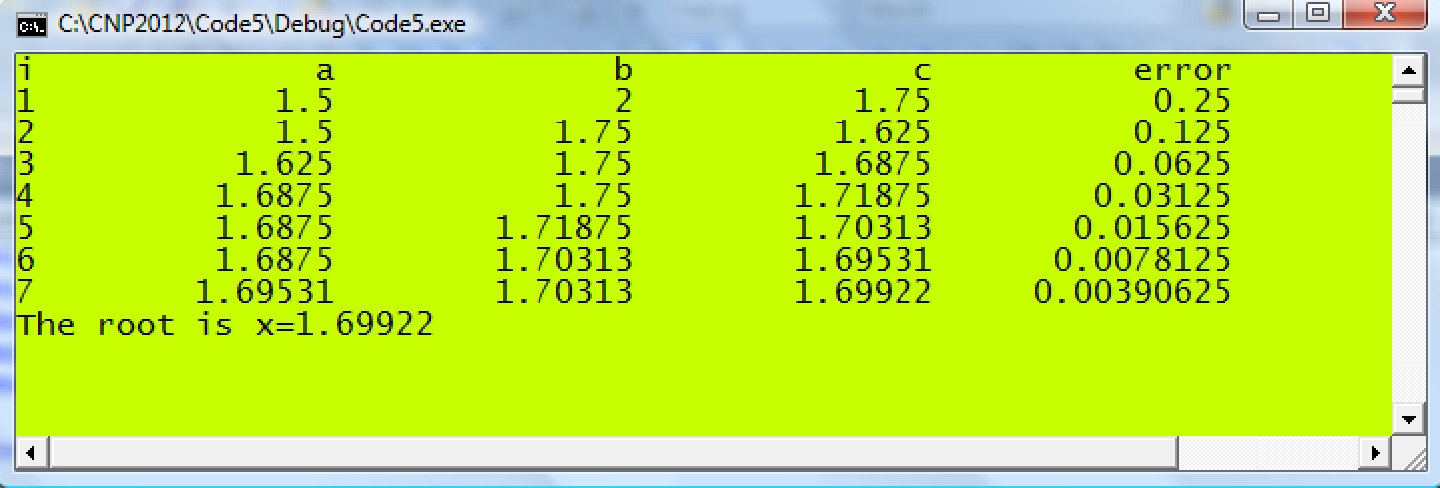
cout << "The root is x=" << c[i] << endl; break;

}

}

delete a,b,c; cin.get();

}



CS2: Sorting Numbers

Swapping numbers

if (p>q)

{

tmp=p; p=q; q=tmp;

}

Example: p=72, q=50

|  |  |  |
| --- | --- | --- |
| if (p>q) |  |  |
| { | *// p=72, tmp=72* |  |
| tmp=p; |  |
| p=q; | *// q=50, p=50* |  |
| q=tmp; | *// tmp=72, q=72* |  |

}

p=50, q=72



**Example 5.1.** Sort the numbers in the list given by 60, 74, 43, 57 and 45 in ascendingorder. The numbers are fully sorted after *k*  3 . The final order from the given list is 43, 45, 57, 60 and 74.

*Solution*

Applying the sorting algorithm above with *k* 1:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | old | | |  | new | |  | i=1: 60<74, no change with w[1]=60, w[2]=74. | |  |
|  | w[1] | | 60 | | |  | 60 |  |  |  |
|  | w[2] | | 74 | | |  | 43 |  |  | i=2: 74>43, swap with w[2]=43, w[3]=74. | |  |
|  | w[3] | | 43 | | |  | 57 |  |  | i=3: 74>57, swap with w[3]=57, w[4]=74. | |  |
|  | w[4] | | 57 | | |  | 45 |  |  | i=4: 74>45, swap with w[4]=45, w[5]=74. | |  |
|  | w[5] | | 45 | | |  | 74 |  |  |  |  |  |
| Continue with *k*  2 : | | | | | | |  |  |  |  |  |  |
|  |  |  | old |  | | | new | |  | i=1: 60>43, swap with w[1]=43, w[2]=60. | |  |
|  | w[1] |  | 60 |  | | | 43 |  |  |  |
|  | w[2] |  | 43 |  | | | 57 |  |  | i=2: 60>57, swap with w[2]=57, w[3]=60. | |  |
|  | w[3] |  | 57 |  | | | 45 |  |  | i=3: 60>45, swap with w[3]=45, w[4]=60. | |  |
|  | w[4] |  | 45 |  | | | 60 |  |  | i=4: 60<74, no change with w[4]=60, w[5]=74. | |  |
|  | w[5] |  | 74 |  | | | 74 |  |  |  |  |  |
| Next iteration with *k*  3 : | | | | | | |  |  |  |  |  |  |
|  |  |  | old | |  | | new |  |  |  | i=1:43<57, no change with w[1]=43, w[2]=57. |  |
|  | w[1] | | 43 | |  | | 43 |  |  |  |  |
|  | w[2] | | 57 | |  | | 45 |  |  |  | i=2: 57>45, swap with w[2]=45, w[3]=57. |  |
|  | w[3] | | 45 | |  | | 57 |  |  |  | i=3:57<60, no change with w[3]=57, w[4]=60. |  |
|  | w[4] | | 60 | |  | | 60 |  |  |  | i=4: 60<74, no change with w[4]=60, w[5]=74. |  |
|  | w[5] | | 74 | |  | | 74 |  |  |  |  |  |

Code5D.cpp: Sorting numbers

#include <iostream>

#include <time.h>

#define N 8

using namespace std;

void main()

{

int v[N+1], w[N+1],tmp, i;

srand(time(0));

for (int i=1;i<=N;i++)

{

v[i]=1+rand()%100; // random numbers from 1 to 100 w[i]=v[i];

}

for (int k=1;k<=N;k++)

for (i=1;i<=N‐1;i++)

if (w[i]>w[i+1]) // swap for low to high

{

tmp=w[i];

w[i]=w[i+1];

w[i+1]=tmp;

}

cout << "the unsorted random numbers v[i] are:" << endl;

for (i=1;i<=N;i++)

cout << v[i] << " ";

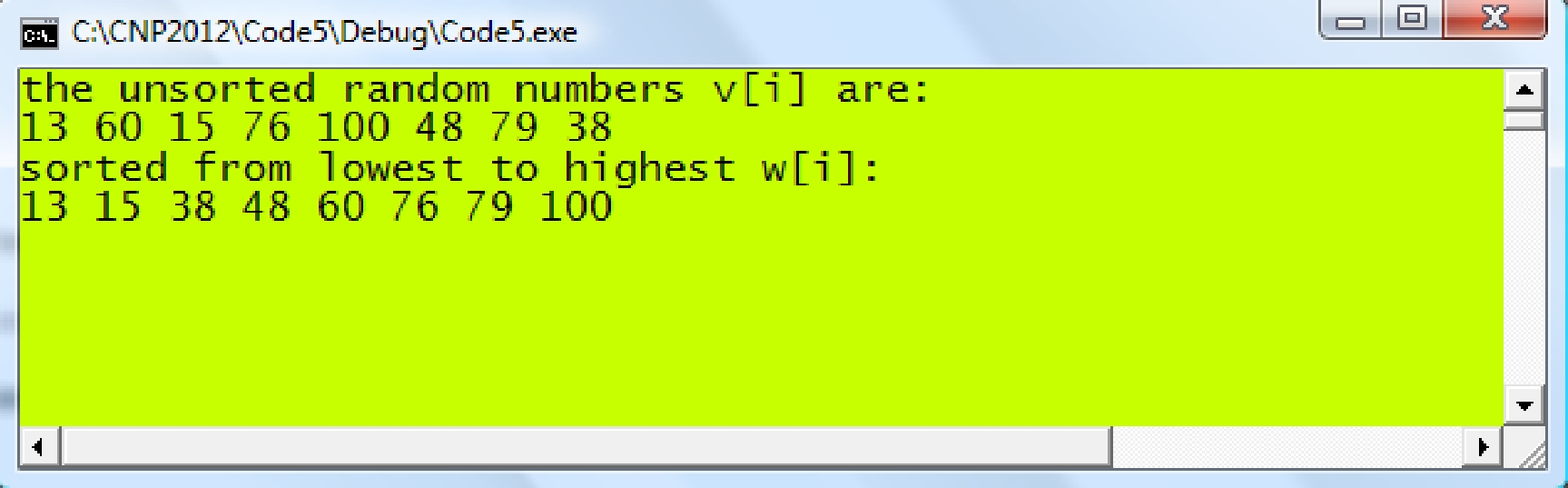
cout<<endl<<"sorted from lowest to highest w[i]:"<<endl; for (i=1;i<=N;i++)

cout << w[i] << " ";

cout << endl;

cin.get();

}



CS3: Detecting the Upper Triangular Matrix





The following code segment determines whether a given matrix is upper triangular method for detection:

*If* flag *is* 0 *then the matrix is not upper triangular.*

*If* flag *is* 1 *then the matrix is upper triangular.*

This code below showing how the input data is read.

ifstream InFile;

InFile.open("Code5E.in");

for (i=1;i<=N;i++)

{

for (j=1;j<=N;j++)

{

InFile >> a[i][j]);

cout << a[i][j] << “ “;

}

cout << endl;

}

Code5E.cpp: Detecting upper triangular matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | Code5E.in |  |  |
| 8 | 0 | 0 | 0 |
| -2 | 1 | 0 | 0 |
| 2 | -1 | 7 | 0 |
| 1 | 8 | 1 | -2 |

#include <iostream>

#include <fstream>

#define N 4

using namespace std;

void main()

{

bool flag=1;

double a[N+1][N+1];

int i,j;

// read matrix A values and display them to confirm

ifstream InFile;

InFile.open("Code5E.in");

for (i=1;i<=N;i++)

{

for (j=1;j<=N;j++)

{

InFile >> a[i][j];

cout << a[i][j] << " ";

}

cout << endl;

}

InFile.close();

// detect the presence of a non‐zero

for (i=2;i<=N;i++)

{

for (int j=1;j<i;j++)

if (a[i][j]!=0)

{

flag=0;

break;

}

if (!flag)

break;

}

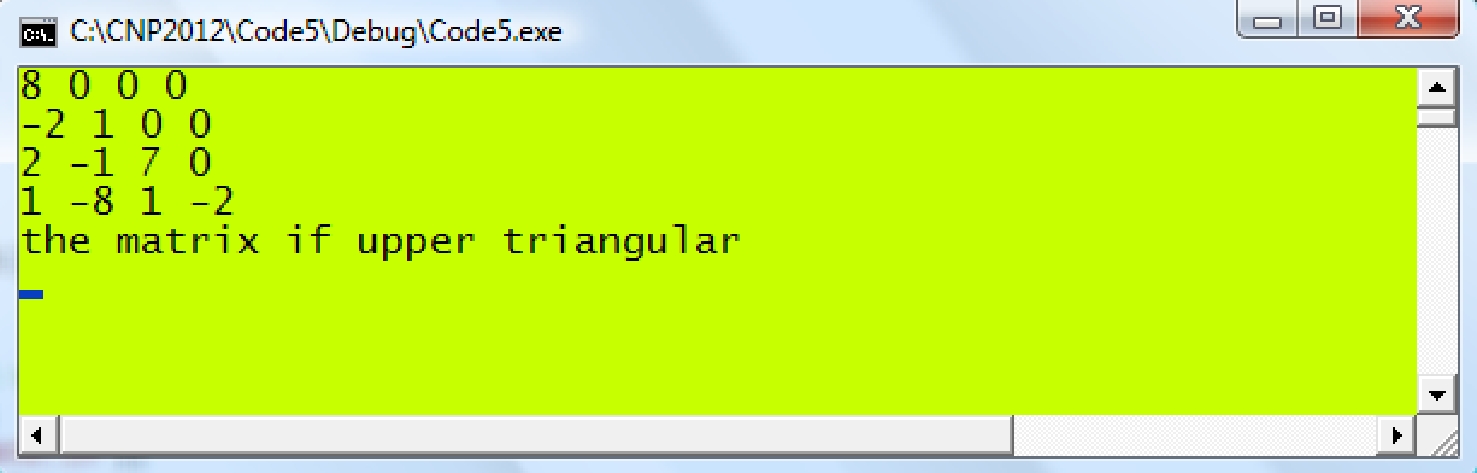
if (flag)

cout<< "the matrix is upper triangular"<<endl;

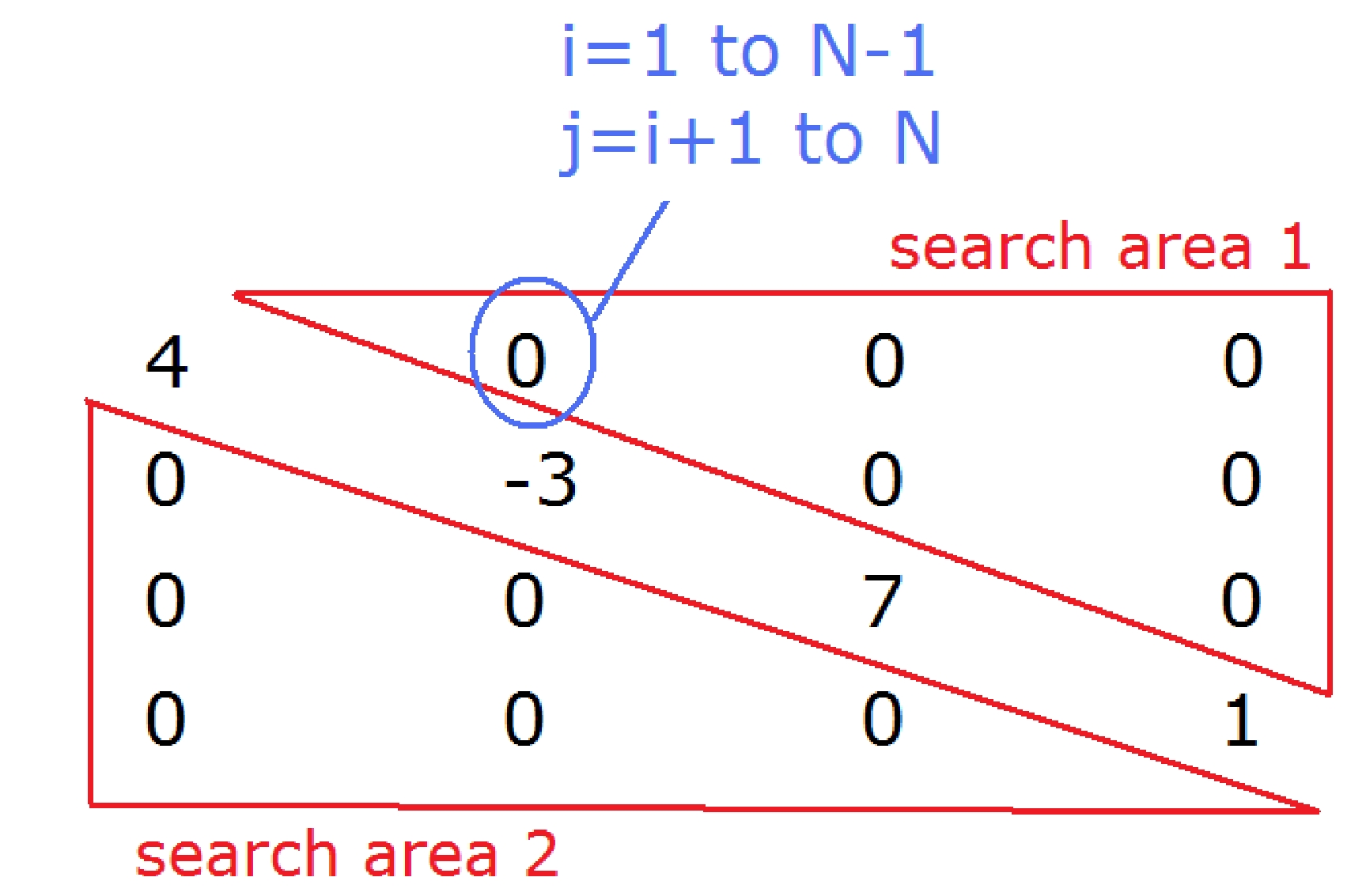
else

cout<< "the matrix is not upper triangular"<< endl;

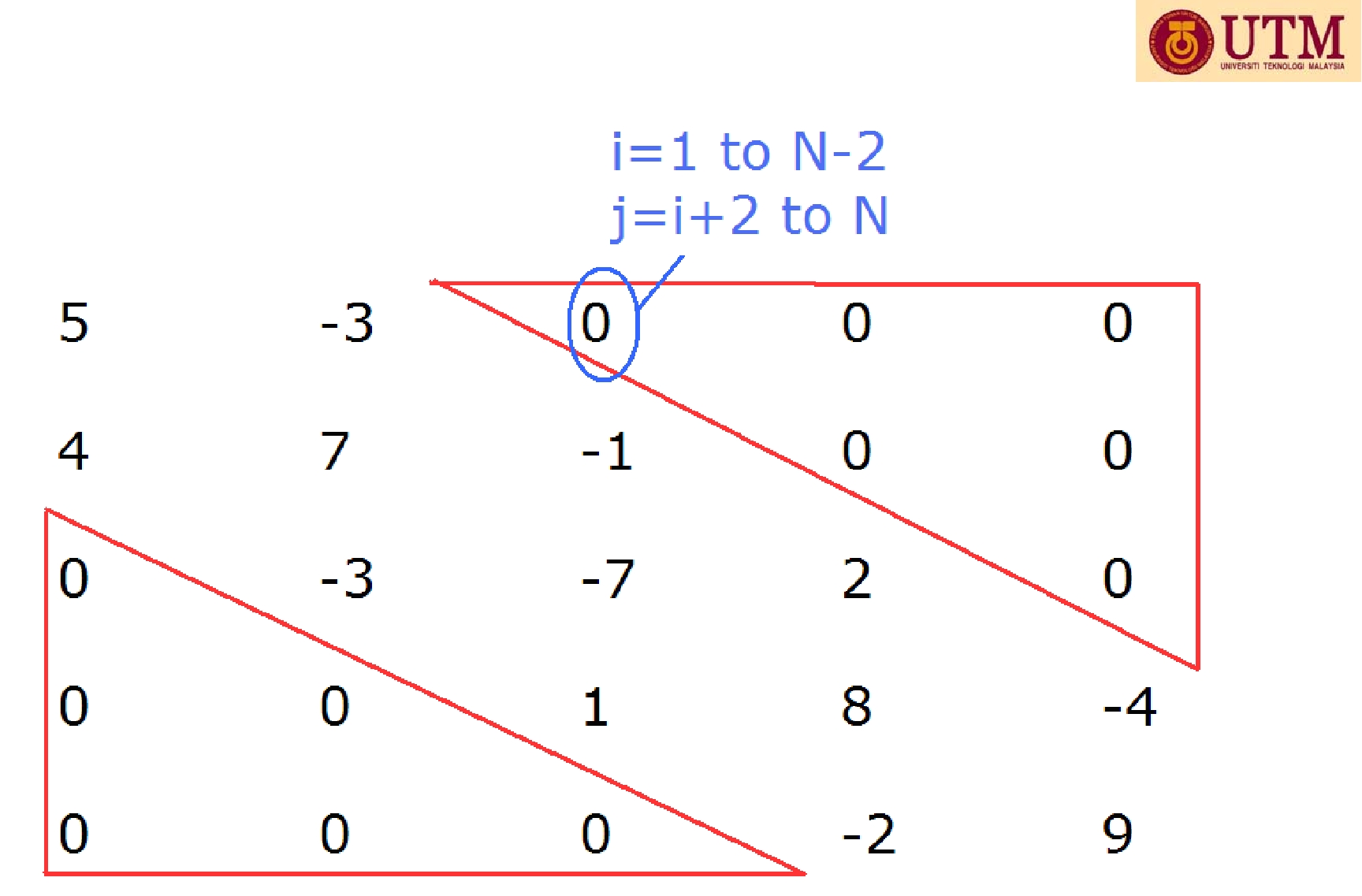
cin.get();

}

**Detecting a diagonal matrix**



**Detecting a tridiagonal matrix**



CS4: Matrix Reduction into an Upper Triangular Form

Method: Row Operations through Gaussian Elimination Method

**Example 5.2.** Find the upper triangular matrix*U*of the following matrix:



|  |  |  |  |
| --- | --- | --- | --- |
| 8.000 | 2.000 | -1.000 | 2.000 |
| 0.000 | 1.500 | -3.250 | -7.500 |
| 0.000 | -1.500 | 7.250 | -1.500 |
| 0.000 | -8.250 | 1.125 | -2.250 |
| *Operations with respect to the second row.* | | |  |
| 8.000 | 2.000 | -1.000 | 2.000 |
| 0.000 | 1.500 | -3.250 | -7.500 |
| 0.000 | 0.000 | 4.000 | -9.000 |
| 0.000 | 0.000 | -16.750 | -43.500 |

*m*  *a*21/ *a*11, *m*  *a*31/ *a*22, *m*  *a*41/ *a*22,

*m*  *a*32/ *a*22, *m*  *a*42/ *a*22,

*a*2 *j* ← *a*2 *j* −*m* \* *a*1 *j a*3 *j* ← *a*3 *j* −*m* \* *a*1 *j a*4 *j* ← *a*4 *j* −*m* \* *a*1 *j*

*a*3 *j* ← *a*3 *j* −*m* \* *a*2 *j a*4 *j* ← *a*4 *j* −*m* \* *a*2 *j*

*Operations with respect to the third row.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8.000 | 2.000 | -1.000 | 2.000 |  |
| 0.000 | 1.500 | -3.250 | -7.500 |  |
| 0.000 | 0.000 | 4.000 | -9.000 |  |
| 0.000 | 0.000 | 0.000 | -81.188 | *m* = *a*43/ *a*33, *a*4 *j* ← *a*4 *j* −*m* \* *a*3 *j* |

General Algorithm for the Row Operations: *n*  4



Row operations on A produce U



Code5F.cpp: Finding the upper triangular matrix

#include <fstream> #include <iostream> #define N 4

using namespace std; void main()

{

int i,j,k; double \*\*a;

a=new double \*[N+1]; for (i=0;i<=N;i++)

a[i]=new double [N+1]; ifstream InFile; InFile.open("Code5F.in",ios::in); for (i=1;i<=N;i++)

for (j=1;j<=N;j++)

InFile >> a[i][j];

InFile.close();

cout << "Matrix A:" << endl;

for (i=1;i<=N;i++)

{

for (j=1;j<=N;j++)

cout << a[i][j] << "\t";

cout << endl;

}

// row operations double m;

for (k=1;k<=N‐1;k++)

for (i=k+1;i<=N;i++)

{

m=a[i][k]/a[k][k]; for (j=1;j<=N;j++)

a[i][j]‐= m\*a[k][j];

}

cout << "Reduced Matrix A after row operations:" << endl; for (i=1;i<=N;i++)

{

for (j=1;j<=N;j++)

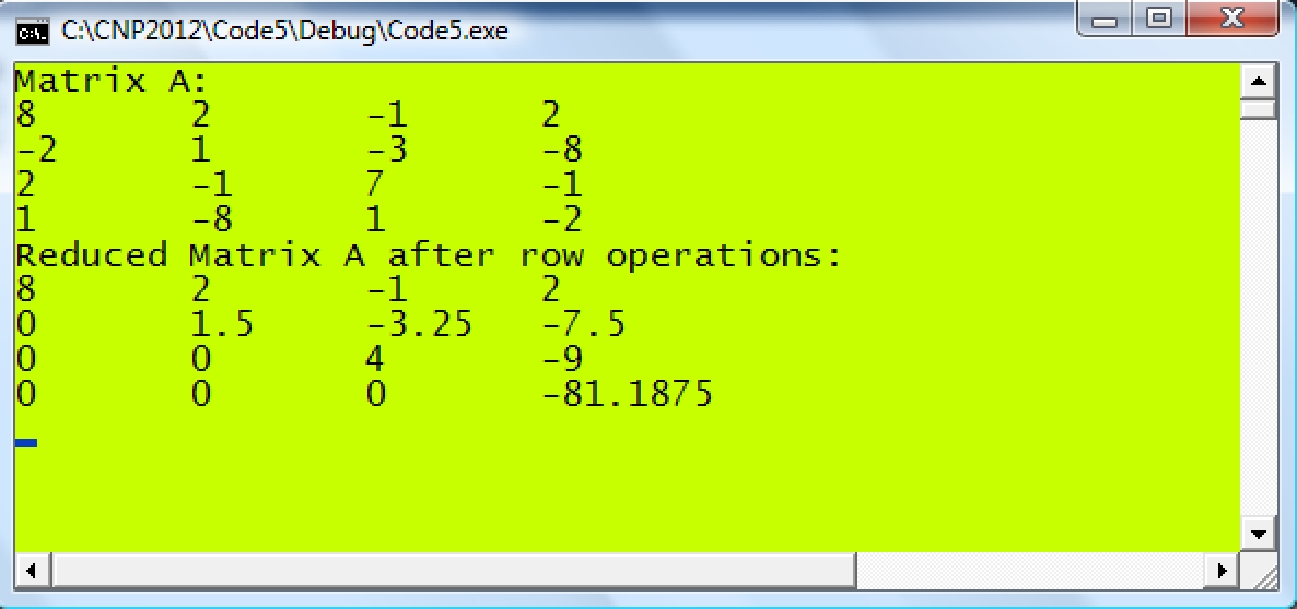
cout << a[i][j] << "\t";

cout << endl;

}

cin.get();

}



**CS5: Computing the Determinant of a Matrix**

**Theorem 5.1.** If a square matrix is reducible to its triangular matrix, the determinant ofthis matrix is the product of the diagonal elements of its reduced upper or lower triangular matrix.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The above | theorem | | | states | | | that if a square matrix | | *A* [*aij* ] | is | reducible to | its |
| upper triangular matrix *U* [*uij* | | | | | ] | | then |  |  |  |  |  |
|  |  | | |  |  | | *N* |  |  |  |  |  |
|  |  | *A* |  |  *U* |  |  | ∏*uii* |  *u*11.*u* 22...*uNN* . |  |  | (5.3) |  |
|  |  | | |  |  | | *i*1 |  |  |  |  |  |

double Product, m; for (k=1;k<=N‐1;k++)

for (i=k+1;i<=N;i++)

{

m=a[i][k]/a[k][k]; for (j=1;j<=N;j++)

a[i][j] ‐= m\*a[k][j];

}

Product=1;

for (i=1;i<=N;i++) Product \*= a[i][i];

Code5G.cpp: Computing the determinant of a matrix

#include <iostream> #include <fstream> #include <iomanip> #define N 4

Using namespace std;

void main()

{

int i,j,k;

double A[N+1][N+1] Product=1,m; cout.setf(ios::fixed); cout.precision(5); cout << "Input matrix A: " << endl; ifstream InFile("Code5F.in");

for (i=1;i<=N;i++)

{

for (j=1;j<=N;j++)

{

InFile >> A[i][j];

cout << setw(10) << A[i][j];

}

cout << endl;

}

InFile.close();

for (k=1;k<=N‐1;k++)

for (i=k+1;i<=N;i++)

{

m=A[i][k]/A[k][k]; for (j=1;j<=N;j++)

A[i][j]‐=m\*A[k][j];

}

cout << endl << "Matrix U:" << endl;

for (i=1;i<=N;i++)

{

for (j=1;j<=N;j++)

cout << setw(10) << A[i][j]; cout << endl;

}

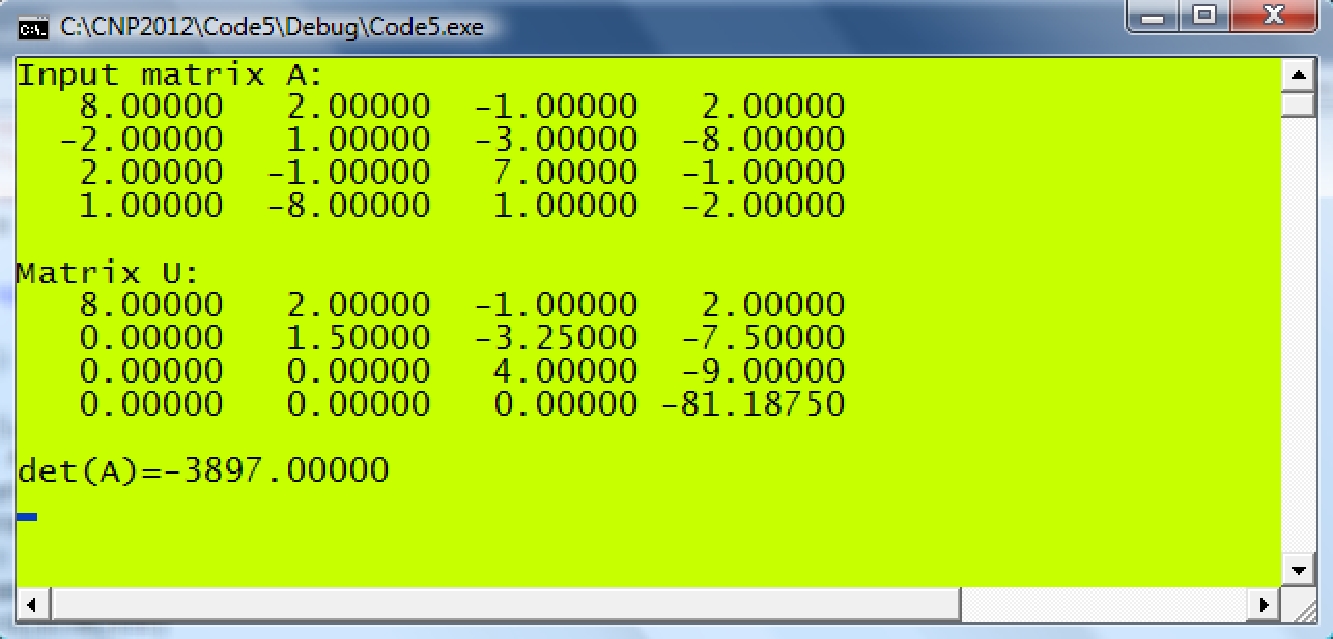
for (i=1;i<=N;i++)

Product \*= A[i][i];

cout << endl << "det(A)=" << Product << endl;

cin.get();

}



MAIN REFERENCE:

Shaharuddin Salleh (2012), C++ Numerical Programming.