UCS545

Lab Assignment I

1. **To generate the random numbers by using different generator parameters and create a file as database to be used for realization of matrix operations.**

**CODE:**

#include <iostream>

#include <fstream>

#include <cstdlib>

using namespace std;

int main()

{

int MaxNumber = 10, random;

ofstream fout;

fout.open("Database.txt");

for (int i = 0; i < 16000000; ++i){

random = rand() % MaxNumber;

fout << random << " ";

}

fout.close();

cout<<"Random Numbers Successfully generated and saved to file.\n";

ifstream fin;

fin.open("Database.txt");

cout<<"A few random numbers inside the database file:\n";

for(int i=0;i<1000;i++){

fin>>random;

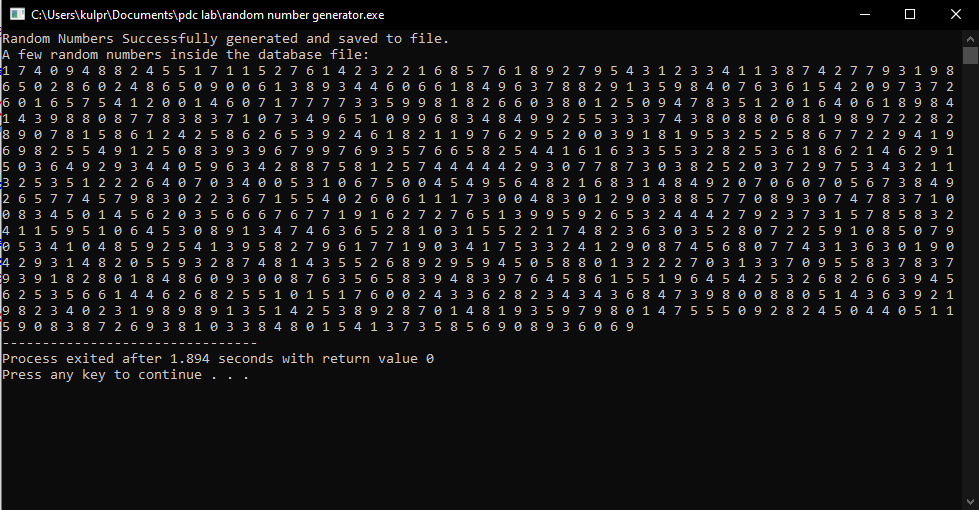
cout<<random<<" ";

}

fin.close();

return 0;

}



1. **Perform the matrix basic operations (Addition, Subtraction Multiply, Matrix Inversion, (a matrix and a vector)), for different dimensions of the matrix.**

**CODE:**

#include <iostream>

#include <fstream>

#include <cmath>

using namespace std;

void Cofactor(int \*\*A, int \*\*temp, int p, int q, int n)

{

int i = 0, j = 0;

for (int row = 0; row < n; row++)

{

for (int col = 0; col < n; col++)

{

if (row != p && col != q)

{

temp[i][j++] = A[row][col];

if (j == n - 1)

{

j = 0;

i++;

}

}

}

}

}

int Determinant(int \*\*A, int n)

{

int D = 0;

if (n == 1)

return A[0][0];

int \*\*temp = new int\*[100];

for (int i = 0; i < 100; ++i)

{

temp[i] = new int[100];

}

int sign = 1;

for (int f = 0; f < n; f++)

{

Cofactor(A, temp, 0, f, n);

D += sign \* A[0][f] \* Determinant(temp, n - 1);

sign = -sign;

}

return D;

}

void adjoint(int \*\*A, int \*\*adj, int N)

{

if (N == 1)

{

adj[0][0] = 1;

return;

}

int sign = 1;

int\*\* temp = new int\* [100];

for (int i = 0; i < 100; ++i)

{

temp[i] = new int[100];

}

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

{

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

{

Cofactor(A, temp, i, j, N);

sign = ((i + j) % 2 == 0) ? 1 : -1;

adj[j][i] = (sign) \* (Determinant(temp, N - 1));

}

}

}

int main()

{

ifstream temp;

temp.open("Database.txt");

const int m = 100;

int\*\* matrixA = new int\* [m];

int\*\* matrixB = new int\* [m];

int\*\* Answer = new int\* [m];

float\*\* InverseMat = new float\* [m];

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

{

matrixA[i] = new int[100];

matrixB[i] = new int[100];

Answer[i] = new int[100];

InverseMat[i] = new float[100];

}

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

{

for (int j = 0; j < m; ++j)

{

matrixA[i][j] = 0;

matrixB[i][j] = 0;

Answer[i][j] = 0;

}

}

int a, r1 = 0, c1 = 0, r2 = 0, c2 = 0, determinant=0;

cout << "What operation would you like to perform?\n1.Addition of Matrices\n2.Subtraction of Matrices\n3.Multipication of Matices\n4.Inverse of a Matrix\n5.Multiplication of a Vector and Matrix" << endl;

cin >> a;

switch (a)

{

case 1: cout << "Enter no. of rows followed by columns(upto 100):" << endl;

cin >> r1 >> c1;

r2 = r1;

c2 = c1;

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

temp >> matrixA[i][j];

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

}

cout << "\n";

}

cout << "\n";

for (int i = 0; i < r2; ++i)

{

for (int j = 0; j < c2; ++j)

{

temp >> matrixB[i][j];

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

}

cout << "\n";

}

cout << "\n";

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

Answer[i][j] = matrixA[i][j]+matrixB[i][j];

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

}

cout << "\n";

}

break;

case 2: cout << "Enter no. of rows followed by columns(upto 100):" << endl;

cin >> r1 >> c1;

r2 = r1;

c2 = c1;

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

temp >> matrixA[i][j];

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

}

cout << "\n";

}

cout << "\n";

for (int i = 0; i < r2; ++i)

{

for (int j = 0; j < c2; ++j)

{

temp >> matrixB[i][j];

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

}

cout << "\n";

}

cout << "\n";

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

Answer[i][j] = matrixA[i][j] - matrixB[i][j];

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

}

cout << "\n";

}

break;

case 3: cout << "Enter no. of rows(Matrix A)(upto 100) followed by columns(Matrix A)(upto 100) followed by columns(Matrix B)(upto 100):" << endl;

cin >> r1 >> c1 >> c2;

r2 = c1;

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

temp >> matrixA[i][j];

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

}

cout << "\n";

}

cout << "\n";

for (int i = 0; i < r2; ++i)

{

for (int j = 0; j < c2; ++j)

{

temp >> matrixB[i][j];

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

}

cout << "\n";

}

cout << "\n";

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c2; ++j)

{

for (int k = 0; k < r2; ++k)

{

Answer[i][j] += matrixA[i][k] \* matrixB[k][j];

}

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

}

cout << "\n";

}

break;

case 4: cout << "Enter the Row = Column size of square matrix:" << endl;

cin >> r1;

c1 = r1;

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

temp >> matrixA[i][j];

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

}

cout << "\n";

}

cout << "\n";

determinant = Determinant(matrixA, r1);

cout <<"Determinant is: " << determinant << endl;

if (determinant == 0)

{

cout << "Inverse does not exist!" << endl;

break;

}

adjoint(matrixA, matrixB, r1);

cout<<"Inverse Matrix is: "<<endl;

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < r1; ++j)

{

InverseMat[i][j] = matrixB[i][j] / (float)determinant;

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

}

cout << "\n";

}

break;

case 5: cout << "Vector into Matrix(enter 1) or Matrix into Vector(enter 2)?" << endl;

int b;

cin >> b;

if (b == 1)

{

cout << "Enter dimensions of the matrix(rows followed by columns):" << endl;

cin >> r2 >> c2;

r1 = 1;

c1 = r2;

}

else if (b == 2)

{

cout << "Enter dimensions of the matrix(rows followed by columns):" << endl;

cin >> r1 >> c1;

r2 = c1;

c2 = 1;

}

else {

cout << "Invalid input!" << endl;

break;

}

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

temp >> matrixA[i][j];

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

}

cout << "\n";

}

cout << "\n";

for (int i = 0; i < r2; ++i)

{

for (int j = 0; j < c2; ++j)

{

temp >> matrixB[i][j];

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

}

cout << "\n";

}

cout << "\n";

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c2; ++j)

{

for (int k = 0; k < r2; ++k)

{

Answer[i][j] += matrixA[i][k] \* matrixB[k][j];

}

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

}

cout << "\n";

}

break;

default: cout << "Invalid Input!" << endl;

break;

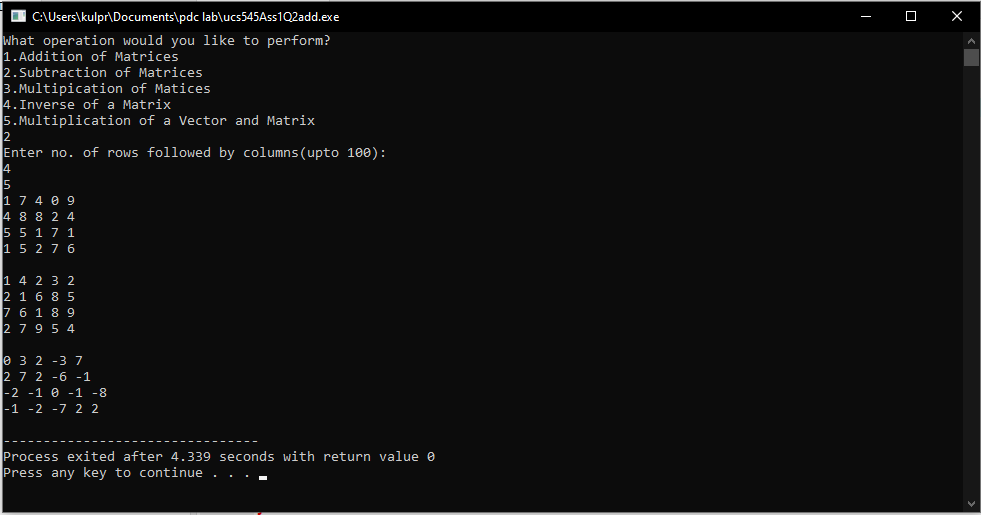
}

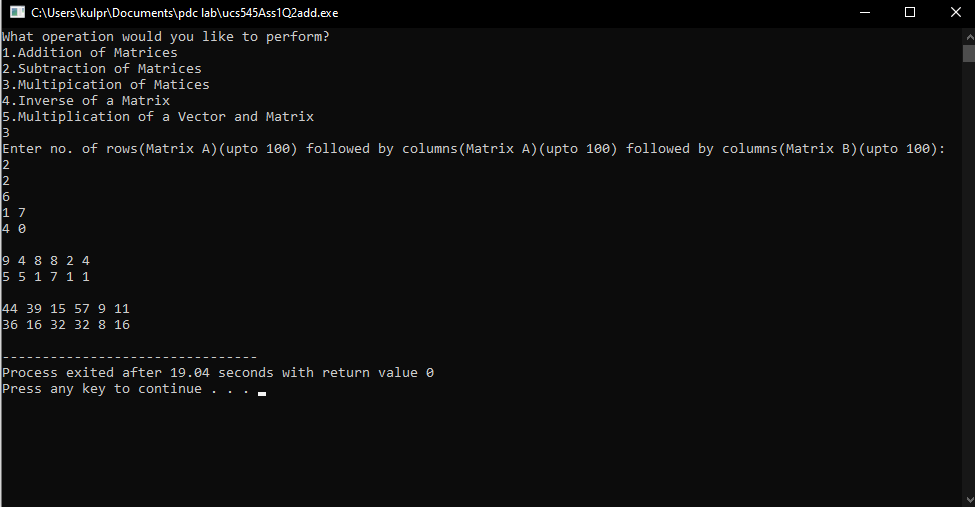
temp.close();

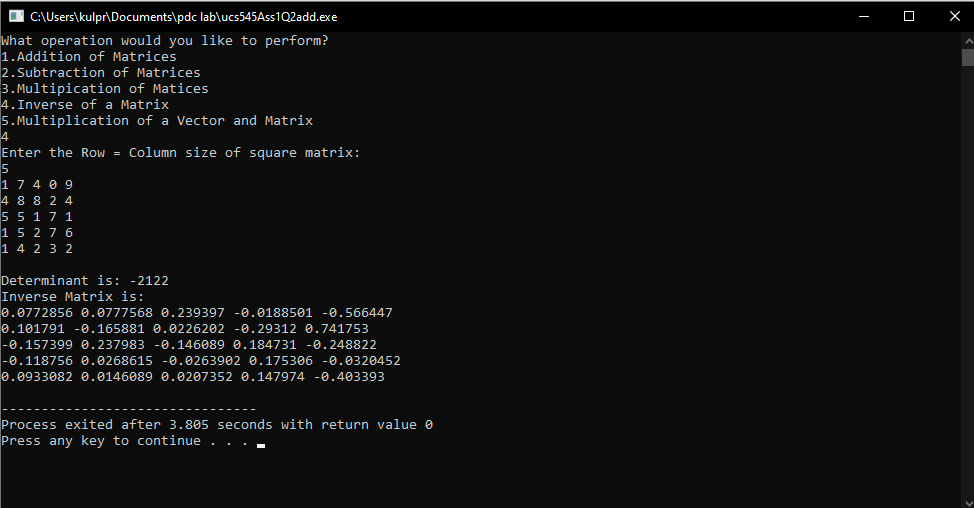
return 0;

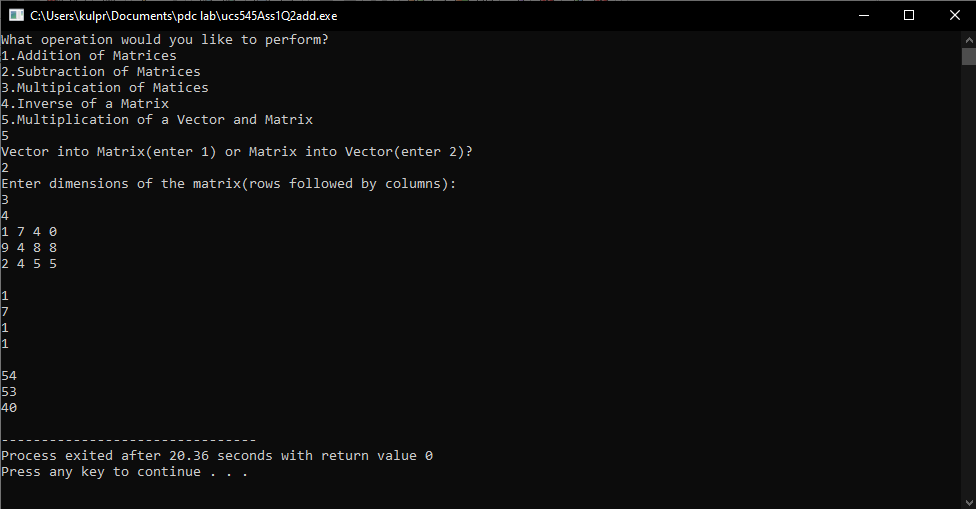
}











1. **Multiply 2 matrices A[2000,2000] and B[2000,6000]. Calculate the computational time.**

**CODE:**

#include<bits/stdc++.h>

#include<fstream>

using namespace std;

int main(){

ifstream temp;

temp.open("Database.txt");

const int m = 2000;

const int n = 2000;

const int p = 6000;

int\*\* a = new int\*[m];

for(int i=0;i<m;i++){

a[i] = new int[n];

for(int j=0;j<n;j++){

temp >> a[i][j];

}

}

int\*\* b = new int\*[n];

for(int i=0;i<n;i++){

b[i] = new int[p];

for(int j=0;j<p;j++){

temp >> b[i][j];

}

}

int\*\* c = new int\*[m];

for(int i=0;i<m;i++){

c[i] = new int[p];

for(int j=0;j<p;j++){

c[i][j] = 0;

}

}

for(int i=0;i<m;i++){

for(int j=0;j<p;j++){

for(int k=0;k<n;k++){

c[i][j] += a[i][k] \* b[k][j];

}

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

}

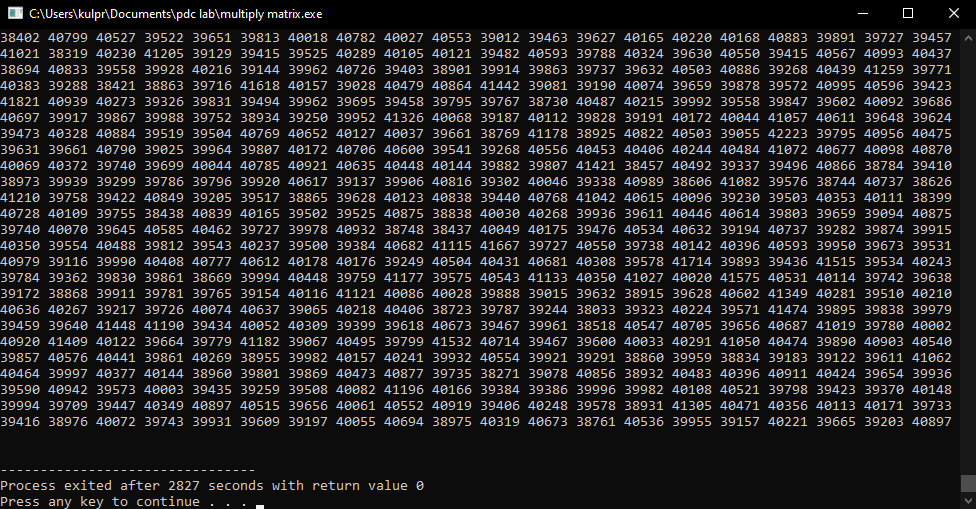
cout<<endl;

}

temp.close();

return 0;

}



Therefore, the **total process execution time/computational time** is **2827 seconds.**

1. **Perform parallel program for the matrix operations (Addition, Multiply, (matrix and vector)). Calculate the computational time.**

**CODE:**

#include <iostream>

#include <fstream>

#include <cmath>

#include<omp.h>

using namespace std;

int main()

{

ifstream temp;

temp.open("Database.txt");

const int m = 100;

int\*\* matrixA = new int\* [m];

int\*\* matrixB = new int\* [m];

int\*\* Answer = new int\* [m];

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

{

matrixA[i] = new int[100];

matrixB[i] = new int[100];

Answer[i] = new int[100];

}

#pragma omp parallel for

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

{

for (int j = 0; j < m; ++j)

{

matrixA[i][j] = 0;

matrixB[i][j] = 0;

Answer[i][j] = 0;

}

}

int a, r1 = 0, c1 = 0, r2 = 0, c2 = 0;

cout << "What operation would you like to perform?\n1.Addition of Matrices\n2.Multipication of Matices\n3.Multiplication of a Vector and Matrix" << endl;

cin >> a;

switch (a)

{

case 1: cout << "Enter no. of rows followed by columns(upto 100):" << endl;

cin >> r1 >> c1;

r2 = r1;

c2 = c1;

#pragma omp parallel for

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

temp >> matrixA[i][j];

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

}

cout << "\n";

}

cout << "\n";

#pragma omp parallel for

for (int i = 0; i < r2; ++i)

{

for (int j = 0; j < c2; ++j)

{

temp >> matrixB[i][j];

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

}

cout << "\n";

}

cout << "\n";

#pragma omp parallel for

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

Answer[i][j] = matrixA[i][j]+matrixB[i][j];

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

}

cout << "\n";

}

break;

case 2: cout << "Enter no. of rows(Matrix A)(upto 100) followed by columns(Matrix A)(upto 100) followed by columns(Matrix B)(upto 100):" << endl;

cin >> r1 >> c1 >> c2;

r2 = c1;

#pragma omp parallel for

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

temp >> matrixA[i][j];

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

}

cout << "\n";

}

cout << "\n";

#pragma omp parallel for

for (int i = 0; i < r2; ++i)

{

for (int j = 0; j < c2; ++j)

{

temp >> matrixB[i][j];

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

}

cout << "\n";

}

cout << "\n";

#pragma omp parallel for

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c2; ++j)

{

for (int k = 0; k < r2; ++k)

{

Answer[i][j] += matrixA[i][k] \* matrixB[k][j];

}

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

}

cout << "\n";

}

break;

case 3: cout << "Vector into Matrix(enter 1) or Matrix into Vector(enter 2)?" << endl;

int b;

cin >> b;

if (b == 1)

{

cout << "Enter dimensions of the matrix(rows followed by columns):" << endl;

cin >> r2 >> c2;

r1 = 1;

c1 = r2;

}

else if (b == 2)

{

cout << "Enter dimensions of the matrix(rows followed by columns):" << endl;

cin >> r1 >> c1;

r2 = c1;

c2 = 1;

}

else {

cout << "Invalid input!" << endl;

break;

}

#pragma omp parallel for

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c1; ++j)

{

temp >> matrixA[i][j];

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

}

cout << "\n";

}

cout << "\n";

#pragma omp parallel for

for (int i = 0; i < r2; ++i)

{

for (int j = 0; j < c2; ++j)

{

temp >> matrixB[i][j];

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

}

cout << "\n";

}

cout << "\n";

#pragma omp parallel for

for (int i = 0; i < r1; ++i)

{

for (int j = 0; j < c2; ++j)

{

for (int k = 0; k < r2; ++k)

{

Answer[i][j] += matrixA[i][k] \* matrixB[k][j];

}

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

}

cout << "\n";

}

break;

default: cout << "Invalid Input!" << endl;

break;

}

temp.close();

return 0;

}

