Rajvaibhav Rahane

17u283 223045

SE-C Comp,Viit,Pune

***CODE:***

/\*

\*

\*@Rajvaibhav Rahane

\*/

/\*

\* Program to perform 2D Matrix operations

\* implements functions

\* create,

\* add,multiply,subtract 2 Matrices,

\* find Transpose and print

\*/

#include<iostream>

using namespace std;

typedef struct Matrix{

int \*matrix;

int rows,columns;

}Matrix;

Matrix createMatrix(int rows,int columns){

Matrix m;

m.matrix=new int[rows\*columns];

m.rows=rows;

m.columns=columns;

cout<<"Enter Elements:\n";

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

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

cin>>\*((m.matrix+i\*columns) + j);

}

}

return m;

}

void printMatrix(Matrix m){

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

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

cout<<\*((m.matrix+i\*m.columns)+ j)<<" ";

cout<<endl;

}

}

Matrix getSumOfMatrices(Matrix m1,Matrix m2){

Matrix sumMatrix;

sumMatrix.matrix=NULL;

if(m1.rows==m2.rows && m1.columns==m2.columns){

sumMatrix.matrix=new int[m1.rows\*m1.columns];

sumMatrix.rows=m1.rows;

sumMatrix.columns=m1.columns;

for(int i=0;i<sumMatrix.rows;i++){

for(int j=0;j<sumMatrix.columns;j++){

\*((sumMatrix.matrix+i\*sumMatrix.columns)+j)=\*((m1.matrix+i\*m1.columns)+j)+\*((m2.matrix+i\*m2.columns)+j);

}

}

return sumMatrix;

}

else{

cout<<"Cannot Add Matrices\n";

}

return sumMatrix;

}

Matrix getDifferenceOfMatrices(Matrix m1,Matrix m2){

Matrix differenceMatrix;

differenceMatrix.matrix=NULL;

if(m1.rows==m2.rows && m1.columns==m2.columns){

differenceMatrix.matrix=new int[m1.rows\*m1.columns];

differenceMatrix.rows=m1.rows;

differenceMatrix.columns=m1.columns;

for(int i=0;i<differenceMatrix.rows;i++){

for(int j=0;j<differenceMatrix.columns;j++){

\*((differenceMatrix.matrix+i\*differenceMatrix.columns)+j)=\*((m1.matrix+i\*m1.columns)+j)-\*((m2.matrix+i\*m2.columns)+j);

}

}

return differenceMatrix;

}

else{

cout<<"Cannot Subtract Matrices\n";

}

return differenceMatrix;

}

Matrix getProductOfMatrices(Matrix m1,Matrix m2){

Matrix productMatrix;

productMatrix.matrix=NULL;

if(m1.columns==m2.rows){

productMatrix.rows=m1.rows;

productMatrix.columns=m2.columns;

productMatrix.matrix=new int[productMatrix.rows\*productMatrix.columns];

for(int i=0;i<productMatrix.rows;i++){

for(int j=0;j<productMatrix.columns;j++){

\*((productMatrix.matrix+i\*productMatrix.columns)+j)=0;

for(int k=0;k<m1.columns;k++){

\*((productMatrix.matrix+i\*productMatrix.columns)+j)+=(\*((m1.matrix+i\*m1.columns)+k)\*(\*(m2.matrix+k\*m2.columns)+j));

}

}

}

return productMatrix;

}

else{

cout<<"Cannot Multiply Matrices\n";

}

return productMatrix;

}

Matrix getTransposeOfMatrix(Matrix m){

Matrix transposeMatrix;

transposeMatrix.matrix=NULL;

if(m.rows==m.columns){

int temp;

transposeMatrix=m;

for(int i=0;i<transposeMatrix.rows-1;i++){

for(int k=i+1;k<m.rows;k++){ //swap elements

temp=\*((transposeMatrix.matrix+i\*transposeMatrix.columns)+k);

\*((transposeMatrix.matrix+i\*transposeMatrix.columns)+k)=\*((transposeMatrix.matrix+k\*transposeMatrix.columns)+i);

\*((transposeMatrix.matrix+k\*transposeMatrix.columns)+i)=temp;

}

}

return transposeMatrix;

}

else{

cout<<"Cannot Find Transpose of given Matrix\n";

}

return transposeMatrix;

}

int main(){

int rows,columns;

cout<<"enter rows and columns";

cin>>rows>>columns;

Matrix m1=createMatrix(rows,columns); //create matrix m1

printMatrix(m1);

cout<<"enter rows and columns";

cin>>rows>>columns;

Matrix m2=createMatrix(rows,columns); //create matrix m2

printMatrix(m2);

Matrix productMatrix=getProductOfMatrices(m1,m2); //find m1\*m2

if(productMatrix.matrix!=NULL){

cout<<"Product Matrix\n";

printMatrix(productMatrix);

}

Matrix sumMatrix=getSumOfMatrices(m1,m2); //find m1+m2

if(sumMatrix.matrix!=NULL){

cout<<"Sum Matrix\n";

printMatrix(sumMatrix);

}

Matrix differencematrix=getDifferenceOfMatrices(m1,m2); //find m1-m2

if(differencematrix.matrix!=NULL){

cout<<"Difference Matrix\n";

printMatrix(differencematrix);

}

Matrix transposeMatrix=getTransposeOfMatrix(m1); //find transpose of m1

if(transposeMatrix.matrix!=NULL){

cout<<"Transpose Matrix\n";

printMatrix(transposeMatrix);

}

return 0;

}

***Output:***

