Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ**

**№4**

з дисципліни:

«ІНФОРМАТИКА 1»

|  |  |
| --- | --- |
|  | Виконав: Черницький Антон  Група: РЕ–11  Викладачі: Катін П.Ю.  (посада, вчене звання, науковий ступінь, прізвище та ініціали)  Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2021

Мета роботи: навчитись працювати з двовимірними масивами, адресами та вказівниками.

Код: #include <stdio.h>

#include <stdlib.h>

#include <time.h>

void mx(int \*\*matrix, int rows, int cols)

{

int i,j;

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

{

for (j = 0; j < cols; j++)

{

printf("%d ", matrix[i][j]);

printf("\t");

}

printf("\n");

}

}

int \*\* memoryAll(int rows, int cols)

{

int i = 0;

int \*\*matrix = (int \*\*)malloc(rows\*sizeof(int \*));

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

{

matrix[i] = (int \*)malloc(cols\*sizeof(int));

}

return matrix;

}

int \*\* matrixInit(int \*\*matrix, int rows, int cols)

{

int i,j,x = 0;

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

{

for( j = 0; j < cols; j++)

{

matrix[i][j] = x;

x += 1;

}

}

return matrix;

}

int \*\* userMatrixInit(int \*\*matrix, int rows, int cols)

{

int i,j,x;

printf("amount of rows = %d\n",rows);

printf("amount of columns = %d\n",cols);

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

{

for( j = 0; j < cols; j++)

{

printf("Element [%d][%d] = ",i+1,j+1);

scanf("%d",&x);

matrix[i][j] = x;

printf("\n");

}

}

return matrix;

}

void sum(int \*\*matrixA, int rowsA, int colsA,int \*\*matrixB, int rowsB, int colsB)

{

int i,j,x;

printf("Sum rows A: \n");

for( i = 0; i < rowsA; i++)

{

x = 0;

for( j = 0; j < colsA; j++)

{

x+=matrixA[i][j];

}

printf("Row [%d] = %d \n",i+1,x);

}

printf("Sum cols B: \n");

for( i = 0; i < colsB; i++)

{

x = 0;

for( j = 0; j < rowsB; j++)

{

x+=matrixB[j][i];

}

printf("Col [%d] = %d \n",i+1,x);

}

}

int Max(int \*\*matrix, int rows, int cols)

{

int i,j,max;

max = matrix[0][0];

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

{

for( j = 0; j < cols; j++)

{

if(max<matrix[i][j])

{

max = matrix[i][j];

}

}

}

return max;

}

int Min(int \*\*matrix, int rows, int cols)

{

int i,j,min;

min = matrix[0][0];

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

{

for(j = 0; j<i; j++)

{

if(matrix[i][j]<min)

{

min = matrix[i][j];

}

}

}

return min;

}

void matrixTran(int \*\*matrix, int rows, int cols)

{

int i = 0,j = 0;

for (i = 0; i < cols; i++)

{

for (j = 0; j < rows; j++)

{

printf("%d ",matrix[j][i]);

printf("\t");

}

printf("\n");

}

}

void matrixMultiplication(int \*\*matrixA, int rowsA, int colsA,int \*\*matrixB, int rowsB, int colsB)

{

int i,j,k,S;

if(rowsA !=rowsB)

{

printf("Unable to multiple\n");

}

else

{

printf("Multiplication matrix A and B (A x B) : \n");

for(i = 0;i<rowsA;i++)

{

for(j = 0;j<colsB;j++)

{

S = 0;

for(k=0; k < rowsB; k++)

{

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

}

printf("%d \t",S);

}

printf("\n");

}

}

}

void clearMemory(int \*\*matrix, int rows)

{

int i;

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

{

free(matrix[i]);

}

free(matrix);

}

int main()

{

int var,rowsA,colsA;

int rowsB,colsB;

printf("A = (N x N)\n");

printf("N = ");

scanf("%d",&rowsA);

printf("\n");

colsA = rowsA;

printf("B = (N x M)\n");

printf("N = ");

scanf("%d",&rowsB);

printf("\n");

printf("M = ");

scanf("%d",&colsB);

printf("\n");

int \*\*A = memoryAll(rowsA,colsA);

int \*\*B = memoryAll(rowsB,colsB);

printf("Did you want to use default matrix? (1 - no, any key - yes)\n");

scanf("%d",&var);

if(var == 1)

{

printf("A : \n");

A = userMatrixInit(A,rowsA,colsA);

printf("B : \n");

B = userMatrixInit(B,rowsB,colsB);

}

else

{

A = matrixInit(A,rowsA,colsA);

B = matrixInit(B,rowsB,colsB);

}

printf("Matrix A: \n");

mx(A,rowsA,colsA);

printf("Matrix B: \n");

mx(B,rowsB,colsB);

printf("Max element in matrix A : %d",Max(A,rowsA,colsA));

printf("\n");

printf("Min element under main diagonal in matrix A : %d",Min(A,rowsA,colsA));

printf("\n");

printf("Transposed Matrix B: \n");

matrixTran(B,rowsB,colsB);

sum(A,rowsA,colsA,B,rowsB,colsB);

matrixMultiplication(A,rowsA,colsA,B,rowsB,colsB);

clearMemory(A,rowsA);

clearMemory(B,rowsB);

return 0;

}

Висновок:

Матриці В (прямокутна), та А (квадратна)

Ми написали програми, що виконують певні дії с массивами – множення, транспонування, знаходження найбільшого числа у певній строчці