Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №4**

з дисципліни: «Інформатика. Основи програмування та алгоритми»

Виконав: Поляков Владислав Сергійович

Ст. групи: РЕ–12

Викладач: доц. Катін П. Ю.

Київ – 2021

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

Код програми:

#include <stdio.h>

#include <stdlib.h>

**double** minelemet(**double**\*\*ptr,**int** n, **int** m){

**double** min=ptr[0][0];

**int** menu;

printf("Choose (1 - main diagonal; 2 - side diagonal)");

scanf("%u",&menu);

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

{

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

{

**if**(menu==1&& i<j && ptr[i][j]<min)

min=ptr[i][j];

**if**(menu==2&& i>j && ptr[i][j]<min)

min=ptr[i][j];

}

}

printf("minimalnu element ");

**if**(menu==1) printf("main diagonal ");

**if**(menu==2) printf("side diagonal ");

printf("%.2lf \n", min);

**return** min;

}

**double** \*\* multmatrix(**double**\*\*ptrA,**int** nA, **int** mA, **double**\*\*ptrB,**int** nB, **int** mB){

**if**(mA!=nB)

{

printf("Error ");

**return** **NULL**;

}

**else**{

**double** \*\*ptrAB;

ptrAB=malloc(nA\***sizeof**(**double**\*));

**for**(**int** i=0; i<nA; i++)

{

ptrAB[i]=malloc(mB\***sizeof**(**double**));

}

**for** (**int** i=0; i<nA; i++)

**for**(**int** j=0; j<mB; j++){

ptrAB[i][j]=0;

**for**(**int** k=0; k<mA; k++)

ptrAB[i][j]+=ptrA[i][k]\*ptrB[k][j];

}

**return** ptrAB;

}

}

**void** printArr(**double**\*\*ptr,**int** n, **int** m){

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

{

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

printf("%7.2lf",ptr[i][j]);

printf("\n");

}

}

**double** maxelemet(**double**\*\*ptr,**int** n, **int** m){

**double** max=ptr[0][0];

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

{

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

**if**(ptr[i][j]>max){

max=ptr[i][j];

}

}

**return** max;

}

**void** transpon(**double**\*\*ptr,**int** n,**int** m, **double**\*\* ptrT)

{

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

{

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

{

ptrT[j][i]=ptr[i][j];

}

}

}

**void** sortrow(**double**\*ptr,**int** n)

{

**double** temp;

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

{

**for**(**int** j=0; j<n-i-1; j++)

{

**if**(ptr[j] > ptr[j+1])

{

temp=ptr[j];

ptr[j]=ptr[j+1];

ptr[j+1]=temp;

}

}

}

}

**void** sumarow(**double**\*\*ptr,**int** n,**int** m)

{

**double** sum;

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

{

sum=0;

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

{

sum+=ptr[i][j];

}

printf("Sum of %u row=%.2lf\n",i,sum);

}

}

**void** sumacol(**double**\*\*ptr,**int** n,**int** m)

{

**double** sum;

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

{

sum=0;

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

{

sum+=ptr[i][j];

}

printf("Sum of %u col=%.2lf\n",j,sum);

}

}

**int** main()

{

**int** nA, nB, mB;

**int** menu, MENU=1;

**double** \*\*ptrA, \*\*ptrB, \*\*ptrT, \*\*ptrAB;

printf("Enter sqMatrix (A) ");

scanf("%u",&nA);

printf("Enter rows (B) ");

scanf("%u",&nB);

printf("Enter colums (B) ");

scanf("%u",&mB);

//----------------------------------------

ptrA=malloc(nA\***sizeof**(**double**\*));

**for**(**int** i=0; i<nA; i++)

{

ptrA[i]=malloc(nA\***sizeof**(**double**));

}

printf("Choose calculation method of (A): 1 - by formula, 2 - manually: ");

scanf("%u",&menu);

**if**(menu==1)

**for**(**int** i=0; i<nA; i++)

{

**for**(**int** j=0; j<nA; j++)

ptrA[i][j]=i+2\*j;

}

**else**

{

printf("Enter elements of the matrix (A)\n");

**for**(**int** i=0; i<nA; i++)

{

**for**(**int** j=0; j<nA; j++)

{

printf("A[%u][%u]= ",i,j);

scanf("%lf",&ptrA[i][j]);

}

}

}

printf("Matrix output (A)\n");

printArr(ptrA,nA,nA);

//----------------------------------------------

ptrB=malloc(nB\***sizeof**(**double**\*));

**for**(**int** i=0; i<nB; i++)

{

ptrB[i]=malloc(mB\***sizeof**(**double**));

}

printf("Choose calculation method of (B): 1 - by formula, 2 - manually: ");

scanf("%u",&menu);

**if**(menu==1)

**for**(**int** i=0; i<nB; i++)

{

**for**(**int** j=0; j<mB; j++)

ptrB[i][j]=2\*i+3\*j;

}

**else**

{

printf("Enter elements of the matrix (B)\n");

**for**(**int** i=0; i<nB; i++)

{

**for**(**int** j=0; j<mB; j++)

{

printf("B[%u][%u]= ",i,j);

scanf("%lf",&ptrB[i][j]);

}

}

}

printf("Matrix output (B)\n");

printArr(ptrB,nB,mB);

**while**(MENU!=0)

{

printf("--------------MENU------------------\n");

printf("1 - searching for max/min element\n 2 - matrix transpose\n 3 - matrix multiplicationA\*B\n ");

printf("4 - Matrix sort (A)\n 5 - Rows' sum (A), Columns' sum (B)\n 0 - Exit the program\n");

printf("...");

scanf("%u",&MENU);

//part 1

**if**(MENU==1){

**double** maxA= maxelemet(ptrA,nA,nA);

printf("Maximum matrix element (A) = %.2lf\n",maxA);

**double** minA=minelemet(ptrA,nA,nA);}

//----------------------------------------------

//part 2

**if**(MENU==2){

ptrT=malloc(mB\***sizeof**(**double**\*));

**for**(**int** i=0; i<mB; i++)

{

ptrT[i]=malloc(nB\***sizeof**(**double**));

}

transpon(ptrB,nB,mB, ptrT);

printf("Transposed matrix (B)\n");

printArr(ptrT,mB,nB);}

//---------------------------

//part 3

**if**(MENU==3){

ptrAB=multmatrix(ptrA,nA,nA,ptrB,nB,mB);

printf("Multiplicated matrices (A\*B)\n");

**if**(ptrAB!=**NULL**)

printArr(ptrAB,nA,mB);}

//-------------------------------

//part 4

**if**(MENU==4){

**int** row=0;

printf("Enter number of a row to sort ");

scanf("%u",&row);

sortrow(ptrA[row],nA);

printf("Sorted matrix\n");

printArr(ptrA,nA,nA);}

//-------------------------------

//part 5

**if**(MENU==5){

printf("The sum of the rows of a matrix (A)\n");

sumarow(ptrA,nA,nA);

printf("The sum of the columns of a matrix (B)\n");

sumacol(ptrB,nB,mB);

}

//-------------------------------

}

**for**(**int** i=0; i<nA; i++)

{

free(ptrA[i]);

}

free(ptrA);

**for**(**int** i=0; i<nB; i++)

{

free(ptrB[i]);

}

free(ptrB);

**if**(ptrT!=0)

{**for**(**int** i=0; i<mB; i++)

{

free(ptrT[i]);

}

free(ptrT);

}

**if**(ptrAB!=0)

{

**for**(**int** i=0; i<nA; i++)

{

free(ptrAB[i]);

}

free(ptrAB);

}

**return** 0;

Висновок: Ми навчилися працювати з динамічними масивами та обчислювати матриці за допомогою масивів.