Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ**

**№3**

з дисципліни:

«ІНФОРМАТИКА. Основи програмування та алгоритми»

|  |  |
| --- | --- |
|  | Виконав: Сорока Михайло  Група: РЕ–11  Викладачі:  доцент Катін П.Ю.  Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2021

Мета роботи: скласти програми для роботи з двовимірними масивами.

Код:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

int \*\* matrix(int \*\*A, int a, int b)//матрица

{

int number;

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

{

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

{

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

scanf("%d",&A[i][j]);

printf("\n");

}

}

return A;

}

int \*\* print\_matrix(int \*\*A, int a, int b )//вывод матрици

{

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

{

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

{

printf("%d \t", A[i][j]);

}

printf("\n");

}

}

int \*\* rand\_matrix(int \*\*A, int a, int b)// рандомные числа матрицы

{

srand(time(NULL));

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

{

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

{

A[i][j] = rand()%51 + 5;

}

}

}

void transpose\_matrix(int \*\*B, int a, int b)//транспортирование матрици B

{

int B1[b][a];

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

{

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

{

B1[j][i] = B[i][j];

}

}

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

{

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

{

printf("%d \t", B1[i][j]);

}

printf("\n");

}

}

void multiplication\_matrix(int \*\*A, int \*\*B, int aA, int aB, int bB)// умножение матрици

{

int C[aA][bB];

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

{

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

{

C[i][j] = 0;

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

C[i][j] += A[i][k] \* B[k][j];

}

}

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

{

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

{

printf("%d \t", C[i][j]);

}

printf("\n");

}

}

void arranging\_numbers\_matrix(int \*\*A, int rowsA, int k)//растановка в порядке возростания

{

int A1[rowsA],temp,A\_max,A\_min,counter = 0;

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

{

if(i==k)

{

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

{

A1[j] = A[i][j];

}

}

}

printf("\n");

temp = A1[0];

A\_min = A1[0];

A\_max = A1[0];

int T;

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

{

if(A1[i] > A\_max)

{

A\_max = A1[i];

}

if(A1[i] < A\_min)

{

A\_min = A1[i];

}

}

printf("\nA(max) = %d\n", A\_max);

printf("\nA(min) = %d\n", A\_min);

printf("row %d\n",k);

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

{

printf("%d \t",A1[i]);

}

printf("\n");

do

{

counter = 0;

for(int i = 0;i<rowsA-1;i++)

{

if(A1[i]>A1[i+1])

{

counter++;

temp = A1[i];

A1[i] = A1[i+1];

A1[i+1] = temp;

}

}

}while(counter !=0);

printf("sorted row %d\n",k);

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

{

printf("%d \t",A1[i]);

}

printf("\n");

}

void diag\_above\_matrix(int \*\*A, int a)//выше диагонали

{

printf("\n\nAbove the main diagonal\n");

int k = 0;

int A1[a];

int A\_max = 0;

int A\_min = 100;

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

{

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

{

if(i < j) {

A1[k] = A[i][j];

printf("A[%d][%d] =%d\t", i, j, A1[k]);

k++;

}

}

}

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

{

if(A1[i] > A\_max)

{

A\_max = A1[i];

}

if(A1[i] < A\_min)

{

A\_min = A1[i];

}

}

printf("\n\nA(max) = %d", A\_max);

printf("\tA(min) = %d", A\_min);

}

void diag\_below\_matrix(int \*\*A, int a)//ниже диагонали

{

printf("\n\nBelow the main diagonal\n");

int k = 0;

int A1[a];

int A\_max = 0;

int A\_min = 100;

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

{

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

{

if(i > j) {

A1[k] = A[i][j];

printf("A[%d][%d] =%d\t", i, j, A1[k]);

k++;

}

}

}

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

{

if(A1[i] > A\_max)

{

A\_max = A1[i];

}

if(A1[i] < A\_min)

{

A\_min = A1[i];

}

}

printf("\n\nA(max) = %d", A\_max);

printf("\tA(min) = %d", A\_min);

}

int \*\* memoryAllocation(int rows, int cols)// выделение памяти

{

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;

}

void clearMemory(int \*\*matrix, int rows)//очистка памяти

{

int i;

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

{

free(matrix[i]);

}

free(matrix);

}

void sum\_rows\_cols(int \*\*matrixA, int rowsA, int colsA,int \*\*matrixB, int rowsB, int colsB)//сумма рядков А та строчек В

{

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 main()// основная функция

{

int rowsA, rowsB, colsA, colsB, b1B, a1B;

int \*\*A;

int \*\*B;

unsigned int variant, end, k;

printf("Enter the size of the square matrix A: ");//Квадратная матрица А

scanf("%d", &rowsA);

colsA=rowsA;

printf("Enter the number of rows matrix B: ");//Матрица В

scanf("%d", &rowsB);

printf("Enter the number of cols matrix B: ");

scanf("%d", &colsB);

A = memoryAllocation(rowsA,colsA);//выделение памяти

B = memoryAllocation(rowsB,colsB);//

printf("\nIf you want to enter from the keyboard - enter \"1\".");// выбор инициализации

printf("\nIf you want it to be asked initialization - enter \"2\".");

printf("\nYour choose: ");

scanf("%d", &variant);

switch(variant)

{

case 1:

{

printf("Your choose to enter from the keyboard");

matrix(A, rowsA, colsA);

matrix(B, rowsB, colsB);

break;

}

case 2:

{

printf("Your choose to be asked initialization");

rand\_matrix(A, rowsA, colsA);

rand\_matrix(B, rowsB, colsB);

break;

}

}

printf ("\nMatrix A\n\n");// вывод матрици A

print\_matrix(A, rowsA, colsA);

printf ("\nMatrix B\n\n");// вывод матрици B

print\_matrix(B, rowsB, colsB);

do//методы

{

printf("\n\nFind the maximum element of matrix A and the minimum of the elements below(above) the main diagonal - enter the \"1\" ");

printf("\n\nTranspose matrix B - enter the \"2\" ");

printf("\n\nFind the matrix products of A and B - enter the \"3\" ");

printf("\n\nSort in ascending order the elements of the array A line from the keyboard - enter the \"4\"");

printf("\n\nDisplay the sum of elements of rows of matrix A and columns of matrix B - enter the \"5\" ");

printf("\n\nEnter the method: ");

scanf("%d", &variant);

switch(variant)

{

case 1://диагонали

{

printf("\nFind the maximum element of matrix A and the minimum of the elements below(above) the main diagonal\n");

diag\_above\_matrix(A, rowsA);

diag\_below\_matrix(A, rowsA);

break;

}

case 2://транспортирование

{

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

transpose\_matrix(B, rowsB, colsB);

break;

}

case 3://умножение А и В

{

printf("\nFind the matrix products of A and B\n\n");

if(rowsA != rowsB)

{

printf("Multiplication is not possible");

}

else

{

multiplication\_matrix(A, B, rowsA, rowsB, colsB);

}

break;

}

case 4://сортировка

{

printf("\nSort in ascending order the elements of the array A line from the keyboard\n");

printf("\nSelect the row in which you want to sort the items. First row - \"k\" = 0, second row - \"k\" = 1 etc");

printf("\nk = ");

scanf("%d", &k);

arranging\_numbers\_matrix(A, rowsA, k);

break;

}

case 5://сумма рядков А та строчек В

{

printf("\nDisplay the sum of elements of rows of matrix A and columns of matrix B\n\n");

sum\_rows\_cols(A, rowsA, colsA, B, rowsB, colsB);

break;

}

}

printf("\nEnter \"1\" if you want to end the program.");

printf("\n\nEnter more than \"1\" if you want to continue the program: ");

scanf("%d", &end);

}while(end>1);

clearMemory(A,rowsA);//очистка памяти

clearMemory(B,rowsB);

return 0;

}