Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра теоретичних основ радіотехніки

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №4**

з дисципліни: «Інформатика 1»

|  |  |
| --- | --- |
|  | Виконав : Філобоченко Ліза  Група: РЕ-11  Викладачі: доцент Катін П.Ю.  Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Підпис: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2021

**Мета роботи**: скласти програми для роботи з двовимірними масивами.

**Ключові моменти**:

1. В лабораторній ми використовуємо вказівники, та вказівники на вказівники
2. Використовуємо оператори malloc для виділення пам'яті.

**Код:**

#include <stdio.h> //деректива припроцесора

#include <stdlib.h>

#include <conio.h>

#include <time.h> // библиотека для генератора рандомных чисел

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

{

int i,j;

for( i=0; i<rows; i++) // цикл для генерации каждого нового рядка "марицы A"

{

for( j=0; j<cols; j++) // цикл для генерации каждого нового элемента в рядке "марицы A"

{

matrix[i][j] = rand()%11;

}

}

return matrix;

}

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

{

int i,j,temp;

for( i=0; i<rows; i++) // цикл для генерации каждого нового рядка "марицы A"

{

for( j=0; j<cols; j++) // цикл для генерации каждого нового элемента в рядке "марицы A"

{

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

scanf("%d",&temp);

matrix[i][j] = temp;

printf("\n");

}

}

return matrix;

}

void Dump(int \*\*matrix,int rows,int cols) //вывод матрици

{

int i,j;

for( i=0; i<rows; i++) // цикл для генерации каждого нового рядка "марицы A"

{

for( j=0; j<cols; j++) // цикл для генерации каждого нового элемента в рядке "марицы A"

{

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

}

printf("\n");

}

printf("\n");

}

void define\_Max\_underMin\_A(int \*\*matrix,int rows)

{

int max=0, imax, jmax, min=100, imin, jmin, varik;

int i,j;

printf(" Enter the way of searching for the smallest element (0 - under the main diagonal, 1 - upper the main diagonal): \n");

scanf("%d", &varik);

// поиск максимального элемента "матрицы А"

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

{

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

{

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

{

max = matrix[i][j];

imax = i;

jmax = j;

}

}

}

// вывод максимального элемента "матрицы А"

printf("\n MATRIX A:");

printf("\n-----------\n MAX element = a [%d %d] = %d", imax+1, jmax+1, max);

switch(varik) //Выбор варианта поиска минимального элемента

{

case 0: // Поиск минимального элемента НИЖЕ главной диагонали

{

for ( i = 0; i < rows; i++) // цикл для поиска в каждом новом рядке

{

for ( j = 0; j < rows; j++) // цикл для перебора каждого нового элемента в рядке

{

if (i > j && matrix[i][j] < min) // если элемент ниже главной диагонали и меньше минимального

{

min = matrix[i][j];

imin = i;

jmin = j;

}

}

}

printf("\n MIN element under main diagonal = a [%d %d] = %d \n\n", imin+1, jmin+1, min);

break;

}

case 1: // Поиск минимального элемента ВЫШЕ главной диагонали

{

for ( i = 0; i < rows; i++) // цикл для поиска в каждом новом рядке

{

for ( j = 0; j < rows; j++) // цикл для перебора каждого нового элемента в рядке

{

if (i < j && matrix[i][j] < min) // если элемент выше главной диагонали и меньше минимального

{

min = matrix[i][j];

imin = i;

jmin = j;

}

}

}

printf("\n MIN element upper main diagonal = a [%d %d] = %d \n\n", imin+1, jmin+1, min);

break;

}

default: // Если неопределён тип поиска минимального элемента

{

printf("\n No such type\n");

exit(0);

}

}

}

void transpon\_B(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 multiplyMatrix(int \*\*matrixA,int rowsA,int colsA,int \*\*matrixB,int rowsB,int colsB) //Перемножение матриц

{

int mult;

int i,j,n;

// если число рядков "матрицы А" не совпадает с числом столбцов "матрицы В"

if(rowsA!=rowsB)

{

printf("\n Enable to find the MULTIPLICATION (na != nb)\n\n");

}

// ищем произведение матриц

else

{

printf("\n MULTIPLICATION of Matrix A and B: \n--------------------------------\n\n");

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

{

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

{

mult=0;

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

{

mult=mult+(matrixA[i][n]\*matrixB[n][j]); // mult+=(matA[i][n]\*matB[n][j]);

}

printf("%5d \t", mult);

}

printf("\n\n\n");

}

}

}

void sort\_Min\_to\_Max(int \*\*matrix,int rows) //сортировка от минимального до мах

{

int \*sort\_matr\_a;

int i,j;

int row\_n, n; // row\_n - номер рядка "матрицы А", n - поточный элемент

printf(" Enter the Row of Matrix A: \n");

scanf("%d", &row\_n);

// выделение памяти под "одномерный масив"

sort\_matr\_a=(int\*)malloc(rows\*sizeof(int));

// цикл для создания одномерного масива

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

{

if (i==row\_n-1)

{

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

{

sort\_matr\_a[j]=matrix[i][j];

}

}

}

// цикл для вывода "одномерного масива"

printf("\n\n %d Row of Matrix A:\n-----------\n", row\_n);

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

{

printf(" %d\n", sort\_matr\_a[i]);

}

// цикл для сортировки рядка "матрицы А" в порядке увеличения элементов

for( i=0; i<rows-1; i++)

{

for( j=0; j<rows-i-1; j++)

{

if(sort\_matr\_a[j]>sort\_matr\_a[j+1])

{

n=sort\_matr\_a[j+1];

sort\_matr\_a[j+1]=sort\_matr\_a[j];

sort\_matr\_a[j]=n;

}

}

}

// цыкл для вывода сортированного рядка "матрицы А"

printf("\n\n SORTED %d Row of Matrix A (MIN -> MAX):\n----------------------------------\n", row\_n);

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

{

printf(" %d\n", sort\_matr\_a[i]);

}

}

void sum\_RowsA\_Cols\_B(int \*\*matrixA,int rowsA,int colsA,int \*\*matrixB,int rowsB,int colsB) //Сума рядоа мат А и сума столбцов мат В

{

int sum;

int i,j;

printf("\n\n SUM OF ROWS of Matrix A: \n-------------------------\n");

// цикл для перебора каждого нового рядка "марицы A"

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

{

// обнулить суму на каждом новом рядке

sum = 0;

// цикл для перебора каждого нового элемента в рядке "марицы А"

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

{

sum+=matrixA[i][j];

}

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

}

printf("\n\n SUM OF COLLONS of Matrix B: \n----------------------------\n");

// цикл для перебора каждого нового столбца "марицы B"

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

{

// обнулить суму на каждом новом столбце

sum = 0;

// цикл для перебора каждого нового элемента в столбце "марицы B"

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

{

sum+=matrixB[j][i];

}

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

}

}

int \*\* memoryAllocation(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;

}

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

{

int i;

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

{

free(matrix[i]);

}

free(matrix);

}

int main()

{

int \*\*matA, \*\*matB;

int rowsA, colsA, rowsB, colsB;

int varik,temp;

srand(time(NULL));

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

scanf("%d",&rowsA);

printf("\n");

colsA=rowsA;

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

scanf("%d",&colsB);

printf("\n");

rowsB=rowsA;

matA = memoryAllocation(rowsA,colsA);

matB = memoryAllocation(rowsB,colsB);

printf("what method : 0 - auto 1 - user\n");

scanf("%d",&temp);

if(temp == 0)

{

matA = autoInit(matA,rowsA,colsA);

matB = autoInit(matB,rowsB,colsB);

}

else if(temp == 1)

{

printf("Matrix A:\n");

userInit(matA,rowsA,colsA);

printf("Matrix B:\n");

userInit(matB,rowsB,colsB);

}

else

{

printf("No such a variant \n");

exit(0);

}

Dump(matA,rowsA,colsA);

Dump(matB,rowsB,colsB);

while(1)

{

printf("\n------------------------ Choose the task: -------------------------\n");

printf(" 1 - Find MAX element of Matrix A and MIN of the elements below (above) the main diagonal Matrix A\n");

printf(" 2 - Transpose Matrix B\n");

printf(" 3 - Find A ? B\n");

printf(" 4 - Sort from MIN to MAX elements of the typed number of row of Matrix A\n");

printf(" 5 - Output the sum of elements in Matrix A Rows and Matrix B Collons\n");

printf(" 0 - Exit from the program\n");

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

scanf("%d", &varik);

switch(varik)

{

case 0:

{

exit(0);

}

case 1:

{

system("cls"); //очистка консоли

printf("Matrix A:\n");

Dump(matA,rowsA,colsA);

define\_Max\_underMin\_A(matA,rowsA);

break;

}

case 2:

{

system("cls");

printf("Matrix B:\n");

Dump(matB,rowsB,colsB);

printf("Transp matrix B:\n");

transpon\_B(matB,rowsB,colsB);

break;

}

case 3:

{

system("cls");

printf("Matrix A:\n");

Dump(matA,rowsA,colsA);

printf("Matrix B:\n");

Dump(matB,rowsB,colsB);

multiplyMatrix(matA,rowsA,colsA,matB,rowsB,colsB);

break;

}

case 4:

{

system("cls");

printf("Matrix A:\n");

Dump(matA,rowsA,colsA);

sort\_Min\_to\_Max(matA,rowsA);

break;

}

case 5:

{

system("cls");

printf("Matrix A:\n");

Dump(matA,rowsA,colsA);

printf("Matrix B:\n");

Dump(matB,rowsB,colsB);

sum\_RowsA\_Cols\_B(matA,rowsA,colsA,matB,rowsB,colsB);

break;

}

default:

clearMemory(matA,rowsA);

clearMemory(matB,rowsB);

break;

}

}

}

**Висновок:** Я навчилася складати програми для роботи з двовимірними масивами.