Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ**

**«РОБОТА З ДВОВИМІРНИМИ МАСИВАМИ»**

з дисципліни: «Інформатика 2»

|  |  |
| --- | --- |
|  | Виконав: Клижевский Денис  Група: РЕ-11  Викладачі: доцент Катін П.Ю.  Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2021

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

Код програми:

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <locale.h>

void output (int \*\*matrix\_A, int side\_square, int \*\*matrix\_B, int height\_rectangular, int width\_rectangular) {

printf("\n\n\t Матрица имеет A вид:\n");

for (int i = 0; i < side\_square; i++) {

printf ("\n");

printf ("\t\t"); for (int j = 0; j < side\_square; j++) printf ("+------+");

printf ("\n\t\t"); for (int j = 0; j < side\_square; j++) printf ("|%5d |", (\*matrix\_A) [i][j]);

printf ("\n\t\t"); for (int j = 0; j < side\_square; j++) printf ("+------+");

}

printf("\n\n\t Матрица имеет Б вид:\n");

for (int i = 0; i < height\_rectangular; i++) {

printf ("\n");

printf ("\t\t"); for (int j = 0; j < width\_rectangular; j++) printf ("+------+");

printf ("\n\t\t"); for (int j = 0; j < width\_rectangular; j++) printf ("|%5d |", (\*matrix\_B) [i][j]);

printf ("\n\t\t"); for (int j = 0; j < width\_rectangular; j++) printf ("+------+");

}

}

void create\_matrix (int \*\*matrix, int height, int width) {

\*matrix = (int \*\*)calloc(height, sizeof(int\*));

if (\*matrix == NULL) {printf ("Error"); exit (0);}

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

(\*matrix)[i]=(int\*)calloc(width, sizeof(int));

if ((\*matrix)[i] == NULL) {printf ("Error"); exit (0);}

}

}

void fill\_matrix (int \*\*matrix, int height, int width, \_Bool mode) {

srand(time(NULL));

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

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

if (mode) (\*matrix) [i][j] = rand()%201-100;

else scanf("%d", &(\*matrix)[i][j]);

printf("\t Матрица имеет вид:\n");

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

printf ("\n");

printf ("\t\t"); for (int j = 0; j < width; j++) printf ("+------+");

printf ("\n\t\t"); for (int j = 0; j < width; j++) printf ("|%5d |", (\*matrix) [i][j]);

printf ("\n\t\t"); for (int j = 0; j < width; j++) printf ("+------+");

}

}

void first\_task (int \*\*matrix\_A, int side\_square) {

int mode;

printf("\n\t\t 1 - Выше");

printf("\n\t\t 2 - Ниже");

printf("\n\t Искать минимум выше или ниже главной диагонали?:\t"); scanf("%d", &mode);

int maximum = -9999, minimum = 9999;

for (int i = 0; i < side\_square; i++)

for (int j = 0; j < side\_square; j++) {

if ((\*matrix\_A) [i][j] > maximum) maximum = (\*matrix\_A) [i][j];

switch (mode) {

case 1: if (i < j) if ((\*matrix\_A) [i][j] < minimum) minimum = (\*matrix\_A) [i][j];

case 2: if (i > j) if ((\*matrix\_A) [i][j] < minimum) minimum = (\*matrix\_A) [i][j];

}

}

printf("\n\t Максимальный элемент матрицы А:\t%d", maximum);

printf("\n\t Минимальный элемент по режиму %d:\t%d", mode, minimum);

}

void second\_task (int \*\*matrix\_B, int \*height\_rectangular, int \*width\_rectangular) {

int \*\*T\_matrix\_B = NULL;

create\_matrix (&T\_matrix\_B, \*width\_rectangular, \*height\_rectangular);

for (int i = 0; i < height\_rectangular; i++)

for (int j = 0; j < width\_rectangular; j++)

T\_matrix\_B[j][i] = (\*matrix\_B)[i][j];

\*matrix\_B = T\_matrix\_B;

int counter = \*height\_rectangular;

\*height\_rectangular = \*width\_rectangular;

\*width\_rectangular = counter;

printf("\t Транспонированая матрица Б имеет вид:\n");

for (int i = 0; i < height\_rectangular; i++) {

printf ("\n");

printf ("\t\t"); for (int j = 0; j < width\_rectangular; j++) printf ("+------+");

printf ("\n\t\t"); for (int j = 0; j < width\_rectangular; j++) printf ("|%5d |", (\*matrix\_B) [i][j]);

printf ("\n\t\t"); for (int j = 0; j < width\_rectangular; j++) printf ("+------+");

}

for (int i =0; i < width\_rectangular; i++) free (T\_matrix\_B[i]); free (T\_matrix\_B);

}

void third\_task (int \*\*matrix\_A, int side\_square, int \*\*matrix\_B, int height\_rectangular, int width\_rectangular) {

if (side\_square != height\_rectangular) {printf ("Error"); exit (0);}

int \*\*matrix\_C = NULL;

create\_matrix (&matrix\_C, side\_square, width\_rectangular);

for (int i = 0; i < side\_square; i++)

for (int j = 0; j < width\_rectangular; j++)

for (int k = 0; k < width\_rectangular; k++)

matrix\_C [i][j] += (\*(matrix\_A) [i][k] \* (\*matrix\_B) [k][j]);

printf("\t Произведение матриц имеет вид:\n");

for (int i = 0; i < height\_rectangular; i++) {

printf ("\n");

printf ("\t\t"); for (int j = 0; j < width\_rectangular; j++) printf ("+------+");

printf ("\n\t\t"); for (int j = 0; j < width\_rectangular; j++) printf ("|%5d |", matrix\_C [i][j]);

printf ("\n\t\t"); for (int j = 0; j < width\_rectangular; j++) printf ("+------+");

}

for (int i =0; i < height\_rectangular; i++) free (matrix\_C[i]); free (matrix\_C);

}

void fourth\_task (int \*\*matrix\_A, int side\_square) {

int mode, counter;

printf("\n\t\t 1 - Столбцы");

printf("\n\t\t 2 - Строчки");

printf("\n\t Сортировать строчку или столбец?:\t"); scanf("%d", &mode);

printf("\n\t Какой именно?:\t"); scanf("%d", &counter);

switch (mode) {

case 1: {

for(int i = 0 ; i < side\_square-1; i++) {

for(int j = 0 ; j < side\_square-i-1 ; j++) {

if((\*matrix\_A) [j][counter] > (\*matrix\_A) [j+1][counter]) {

int pointer = (\*matrix\_A) [j][counter];

(\*matrix\_A) [j][counter] = (\*matrix\_A) [j+1][counter];

(\*matrix\_A) [j+1][counter] = pointer;

}}}} break;

case 2: {

for(int i = 0 ; i < side\_square-1; i++) {

for(int j = 0 ; j < side\_square-i-1 ; j++) {

if((\*matrix\_A) [counter][j] > (\*matrix\_A) [counter][j+1]) {

int pointer = (\*matrix\_A) [counter][j];

(\*matrix\_A) [counter][j] = (\*matrix\_A) [counter][j+1] ;

(\*matrix\_A) [counter][j+1] = pointer;

}}}} break;

}

printf("\t Матрица имеет вид:\n");

for (int i = 0; i < side\_square; i++) {

printf ("\n");

printf ("\t\t"); for (int j = 0; j < side\_square; j++) printf ("+------+");

printf ("\n\t\t"); for (int j = 0; j < side\_square; j++) printf ("|%5d |", (\*matrix\_A) [i][j]);

printf ("\n\t\t"); for (int j = 0; j < side\_square; j++) printf ("+------+");

}

}

void fifth\_task (int \*\*matrix\_A, int side\_square, int \*\*matrix\_B, int height\_rectangular, int width\_rectangular) {

int \*summ\_A;

create\_matrix (&summ\_A, side\_square, 1);

int \*summ\_B;

create\_matrix (&summ\_B, width\_rectangular, 1);

for (int i = 0; i < side\_square; i++)

for (int j = 0; j < side\_square; j++)

summ\_A [i] += (\*matrix\_A) [i][j];

for (int j = 0; j < height\_rectangular; j++)

for (int i = 0; i < width\_rectangular; i++)

summ\_B [j] += (\*matrix\_B) [i][j];

printf("\n\t Сумма элементов строк матрицы А:\n");

for (int i = 0; i < side\_square; i++) printf("\t %d", summ\_A [i]);

printf("\n\t Сумма элементов рядков матрицы B:\n");

for (int i = 0; i < width\_rectangular; i++) printf("\t %d", summ\_B [i]);

}

void process (int \*\*matrix\_A, int \*side\_square, int \*\*matrix\_B, int \*height\_rectangular, int \*width\_rectangular, \_Bool \*mode, int \*number) {

printf("\n\n\n\t\t 1 - Найти максимальный элемент матрицы А и\n\t\t\tминимальный из элементов, находящихся ниже (выше) главной диагонали");

printf("\n\t\t 2 - Транспонировать матрицу Б");

printf("\n\t\t 3 - Найти матричное произведение АхБ");

printf("\n\t\t 4 - Отсортировать по возрастанию элементы\n\t\t\tзаданного с клавиатуры строки массива А (индексация начинается с нуля)");

printf("\n\t\t 5 - Вывести на экран сумму элементов строк матрицы А и столбцов матрицы Б");

printf("\n\t\t 0 - Вывести на экран матрицы А и Б");

printf("\n\t Введите номер желаемой задачи:\t"); scanf("%d", &(\*number));

switch (\*number) {

case 1: first\_task (matrix\_A, \*side\_square); break;

case 2: second\_task (matrix\_B, height\_rectangular, width\_rectangular); break;

case 3: third\_task (matrix\_A, \*side\_square, matrix\_B, \*height\_rectangular, \*width\_rectangular); break;

case 4: fourth\_task (matrix\_A, \*side\_square); break;

case 5: fifth\_task (matrix\_A, \*side\_square, matrix\_B, \*height\_rectangular, \*width\_rectangular); break;

case 0: output (matrix\_A, \*side\_square, matrix\_B, \*height\_rectangular, \*width\_rectangular); break;

default: process(matrix\_A, side\_square, matrix\_B, height\_rectangular, width\_rectangular, mode, number);

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

int side\_square, height\_rectangular, width\_rectangular, number;

\_Bool mode, end;

printf("\n\n\t++===================================================++\n");

printf("\t|| №5 ||\n");

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

printf("\t|| Поработать с динамическими массивами ||\n");

printf("\t++===================================================++\n\n");

printf("\n\t\t 0 - Нет");

printf("\n\t\t 1 - Да");

printf("\n\t Заполнить массив случано?:\t"); scanf("%d", &mode);

printf("\n\t Введите размер стороны квадратной матрицы А\t"); scanf("%d", &side\_square);

int \*\*matrix\_A = NULL;

create\_matrix (&matrix\_A, side\_square, side\_square);

fill\_matrix (&matrix\_A, side\_square, side\_square, mode);

printf("\n\n\t Введите высоту прямоугольной матрицы В\t"); scanf("%d", &height\_rectangular);

printf("\t Введите ширину прямоугольной матрицы В\t"); scanf("%d", &width\_rectangular);

int \*\*matrix\_B = NULL;

create\_matrix (&matrix\_B, height\_rectangular, width\_rectangular);

fill\_matrix (&matrix\_B, height\_rectangular, width\_rectangular, mode);

process(&matrix\_A, &side\_square, &matrix\_B, &height\_rectangular, &width\_rectangular, &mode, &number);

printf("\n\n\n\t\t 1 - Да");

printf("\n\t Завершить выполнени программы?:\t"); scanf("%d", &end);

while (!end)

process(&matrix\_A, &side\_square, &matrix\_B, &height\_rectangular, &width\_rectangular, &mode, &number);

return 0;

}