Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра теоретичних основ радіотехніки

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №4**

з дисципліни: «Інформатика 1»

|  |  |
| --- | --- |
|  | Виконав: Демчук В’ячеслав Андрійович  Група: РЕ-11  Викладачі: доцент Катін П.Ю.  Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Підпис: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2021

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

**Код програми:**

**#define true 0**

**#define false 1**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**typedef int bool;**

**void matrix\_multiplication(int\*\*\* matrixA, int\*\*\* matrixB, int\*\*\* matrixC, int rowsA, int colsB);**

**void matrix\_transponate(int\*\*\* matrix, int\*\*\* matrixT, int\* rows, int\* cols);**

**int matrix\_minVal(int\*\* matrix, int rows, int cols);**

**int matrix\_maxVal(int\*\* matrix, int rows, int cols);**

**void matrix\_print(int\*\* matrix, int rows, int cols);**

**void matrix\_fill(int\*\* matrix, int rows, int cols);**

**void matrix\_setParams(int\* a\_rows, int\* a\_cols, int\* b\_rows, int\* b\_cols);**

**void matrix\_sumRows(int\*\* matrix, int rows, int cols);**

**void matrix\_sumCols(int\*\* matrix, int rows, int cols);**

**void matrix\_eraseMem(int\*\* matrix, int cols);**

**void matrix\_sort(int\*\* matrix, int cols);**

**int matrix\_init(int rows, int cols);**

**int matrix\_TMx(int rows, int cols);**

**int set\_function();**

**void header();**

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

**{**

**int\*\* mTMx = (int)calloc(rows, sizeof(int));**

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

**mTMx[i] = (int\*)calloc(cols, sizeof(int));**

**return mTMx;**

**}**

**void header()**

**{**

**printf("\t\t\t\t\tРобота з динамiчними масивами\n");**

**printf("\t\t\t\t\t Лабораторна робота №4\n\t\t\t\t\t Демчук В'ячеслав РЕ-11\n");**

**printf("Натиснiть будь-яку кнопку для продовження...\r");**

**getch();**

**system("cls");**

**printf("\t\t\t\t\tРобота з динамiчними масивами\n");**

**}**

**int set\_function()**

**{**

**int function;**

**printf("1 -- Максимальний елемент матрицi;\n"**

**"2 -- Транспонувати матрицю B;\n"**

**"3 -- Знайти матричний добуток AxB;\n"**

**"4 -- Вiдсортувати за зростанням елементи заданого з клавiатури рядка масиву А;\n"**

**"5 -- Вивести на екран суму елементiв рядкiв матрицi А та стовпцiв матрицi B;\n"**

**"Виберiть дiю з матрицями: ");**

**scanf("%d", &function);**

**switch (function)**

**{**

**case 1:**

**return 1;**

**case 2:**

**return 2;**

**case 3:**

**return 3;**

**case 4:**

**return 4;**

**case 5:**

**return 5;**

**default:**

**printf("Спробуйте знову!!!\n");**

**set\_function();**

**}**

**}**

**matrix\_multiplication(int\*\*\* matrixA, int\*\*\* matrixB, int\*\*\* matrixC, int rowsA, int colsB)**

**{**

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

**{**

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

**{**

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

**{**

**(\*matrixC)[i][j] += (\*matrixA)[i][k] \* (\*matrixB)[k][j];**

**}**

**}**

**}**

**matrix\_print(matrixC, rowsA, colsB);**

**}**

**void matrix\_transponate(int\*\*\* matrix, int\*\*\* matrixT, int\* rows, int\* cols)**

**{**

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

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

**(\*matrixT)[j][i] = (\*matrix)[i][j];**

**}**

**}**

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

**free((\*matrix)[i]);**

**}**

**free(\*matrix);**

**int tmp = \*rows;**

**\*rows = \*cols;**

**\*cols = tmp;**

**\*matrix = \*matrixT;**

**printf("Транспонована матриця B:\n");**

**matrix\_print(\*matrix, \*rows, \*cols);**

**}**

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

**{**

**int min, i, j, diagonal;**

**printf("\n1 -- Знайти мiнiмальне значення над головною дiагоналлю;\n"**

**"2 -- Знайти мiнiмальне значення пiд головною дiагоналлю;\n"**

**"Зробiть свiй вибiр: ");**

**scanf("%d", &diagonal);**

**if (diagonal == 1)**

**{**

**i = 0, j = 0;**

**min = matrix[i][j];**

**while (i < rows)**

**{**

**j = i;**

**while (j < cols)**

**{**

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

**min = matrix[i][j];**

**j++;**

**}**

**i++;**

**}**

**}**

**if (diagonal == 2)**

**{**

**i = rows - 1, j = rows - 1;**

**min = matrix[i][j];**

**while (i >= 0)**

**{**

**j = i;**

**while (j >= 0)**

**{**

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

**min = matrix[i][j];**

**j--;**

**}**

**i--;**

**}**

**}**

**printf("Мiнiмальне значення :%d\n", min);**

**return min;**

**}**

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

**{**

**int max = 0;**

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

**{**

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

**{**

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

**max = matrix[i][j];**

**}**

**}**

**printf("Максимальне значення :%d", max);**

**return max;**

**}**

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

**{**

**printf("Матричний вигляд: \n");**

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

**{**

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

**{**

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

**}**

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

**}**

**}**

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

**{**

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

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

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

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

**}**

**}**

**}**

**void matrix\_setParams(int\* a\_rows, int\* a\_cols, int\* b\_rows, int\* b\_cols)**

**{**

**printf("Введiтьт розмiри матрицi А: ");**

**scanf("%d", a\_rows); \*a\_cols = \*a\_rows;**

**printf("Введiтьт к-ть стовпцiв матрицi B: ");**

**scanf("%d", b\_cols);**

**printf("Введiтьт к-ть рядкiв матрицi B: ");**

**scanf("%d", b\_rows);**

**}**

**void matrix\_eraseMem(int\*\* matrix, int cols)**

**{**

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

**{**

**free(matrix[i]);**

**}**

**free(matrix);**

**}**

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

**{**

**int Sum = 0;**

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

**{**

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

**{**

**Sum += matrix[i][j];**

**}**

**}**

**printf("Сума рядкiв = %d\n", Sum);**

**}**

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

**{**

**int Sum = 0;**

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

**{**

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

**{**

**Sum += matrix[i][j];**

**}**

**}**

**printf("Сума стовпцiв = %d\n", Sum);**

**}**

**void matrix\_sort(int\*\* matrix, int cols)**

**{**

**int row = 0, tmp = 0;**

**printf("Який рядок ви хочете вiдсортувати? (к-ть рядкiв %d)",cols);**

**scanf("%d", &row);**

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

**for (int j = 0; j < cols - 1; j++) {**

**if (matrix[row][j] > matrix[row][j + 1]) {**

**tmp = matrix[row][j];**

**matrix[row][j] = matrix[row][j + 1];**

**matrix[row][j + 1] = tmp;**

**}**

**}**

**}**

**matrix\_print(matrix, cols, cols);**

**}**

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

**{**

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

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

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

**return matrix;**

**}**

**int main()**

**{**

**setlocale(0, "");**

**bool exit = true;**

**int a\_rows, a\_cols, b\_rows, b\_cols;**

**int A\_matrix, B\_matrix, C\_matrix, function;**

**header();**

**matrix\_setParams(&a\_rows, &a\_cols, &b\_rows, &b\_cols);**

**int\*\* matrixT = matrix\_TMx(b\_cols, b\_rows);**

**A\_matrix = matrix\_init(a\_rows, a\_cols);**

**B\_matrix = matrix\_init(b\_rows, b\_cols);**

**printf("Заповнiть матрицю A: \n");**

**matrix\_fill(A\_matrix, a\_rows, a\_cols);**

**matrix\_print(A\_matrix, a\_rows, a\_cols);**

**printf("Заповнiть матрицю B: \n");**

**matrix\_fill(B\_matrix, b\_rows, b\_cols);**

**matrix\_print(B\_matrix, b\_rows, b\_cols);**

**function = set\_function();**

**while (exit == true)**

**{**

**switch(function)**

**{**

**case 1:**

**matrix\_maxVal(A\_matrix, a\_rows, a\_cols);**

**matrix\_minVal(A\_matrix, a\_rows, a\_cols);**

**break;**

**case 2:**

**matrix\_transponate(&B\_matrix, &matrixT, &b\_rows, &b\_cols);**

**break;**

**case 3:**

**if (a\_cols == b\_rows) {**

**matrix\_multiplication(&A\_matrix, &B\_matrix, &C\_matrix, a\_rows, b\_cols);**

**}**

**break;**

**case 4:**

**matrix\_sort(A\_matrix, a\_cols);**

**break;**

**case 5:**

**matrix\_sumRows(A\_matrix, a\_rows, a\_cols);**

**matrix\_sumCols(B\_matrix, b\_rows, b\_cols);**

**break;**

**}**

**printf("Завершити виконання програми (press 1, if no press any other key)");**

**scanf("%d", &exit);**

**if(exit != false)**

**{**

**exit = true;**

**}**

**if (exit == false)**

**{**

**matrix\_eraseMem(A\_matrix, a\_cols);**

**matrix\_eraseMem(B\_matrix, b\_cols);**

**matrix\_eraseMem(C\_matrix, b\_cols);**

**matrix\_eraseMem(matrixT, b\_rows);**

**return 0;**

**}**

**else**

**{**

**function = set\_function();**

**}**

**}**

**}**