Національний технічний університет України

# «Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра теоретичних основ радіотехніки

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ** №**4**

з дисципліни: «Інформатика 1»

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Виконав:Луцкевич Володимир Андрійович |  |
| Група: РЕ-12  Викладачі: доцент Катін П.Ю.  Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Підпис: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2021

Мета: Скласти алгоритм для кожної задачі обробки двовимірних масивів.

Код:

#include <stdio.h>

#include <conio.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include < malloc.h > // для використання функцій динамічного розподілу пам'яті

#include <conio.h> // для створення текстовоо інтерфесу

#include <time.h>

#define size 8 //розмір масиву

void printArray( int[], int); // за допомогою цієї функцій виводимо значення заданого масиву

int search\_Max(int[], int); // функція в якій вказано масив і розмір масиву, повертає максимально знайдене значення цього масиву

int search\_Min(int[], int);

void search\_Max2(int[], int, int \* pMax, int \* pIndex); // pMax- зберігає адресу зміінної Max2, pIndex- зберігає адресу зміної index

void sort\_1D(int \*z, int S); //сортування одномірного масиву z, S - розмір масиву

int ArrayBFunc(int[size], int); // масив складається із 4 елементів

//void printArray2(int\*, int);

//void printArray3(int\*, int);

int main()

{

int num1;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int\* a; // вказівник на масив

int i, j, n, m; // індекси: а - вказівник на масив, m - кількість стовпців i - індект рядка j індекс стовпця

system("chcp 1251");

system("cls");

printf("Number of lines: ");

scanf\_s("%d", &n);

printf("Number of columns: ");

scanf\_s("%d", &m);

//malloc - функція для визначення розміру масиву в байтах

// int sizeof() - для точного визначення розміру елементу

// виділення пам'яті

a = (int\*)malloc(n \* m \* sizeof(int));// n·m·(розмір елементу)- об'єм пам'ті необхідний для розміщення двовимірного масиву

//printf("%d Розмір масиву\n\t = ", a);

// ввести кожен елемент масиву

for (i = 0; i < n; i++) // цикл по рядкам

{

for (j = 0; j < m; j++) // цикл по стовпцям

{

printf("a[%d][%d] = ", i, j);//index = i\*m+j;

scanf\_s("%d", (a + i \* m + j)); // а - вказівник на масив, m - кількість стовпців i - індект рядка j індект стовпця

}

}

// вивести кожен елемент масиву

for (i = 0; i < n; i++) // цикл по рядкам

{

for (j = 0; j < m; j++) // цикл по стовпцям

{

//\*(a + i \* m + j) звернення до елементу index = i\*m+j; // кожен елмен

printf("%5d ", \*(a + i \* m + j)); // поле шириною 5 символів під елмент масиву

}

printf("\n");

}

int ArrayA = \*(a + i \* m + j);

num1 = \*(a + 0 \* m + 1); // змінюючи і та j можна викликати будь який елемент масиву

printf("\nTransposed matrix A\n");

for (i = 0; i < n; i++)// ідентична частина, але результат виводиться в консоль

{

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

{

int ArrayAT = \*(a + j \* m + i);

printf("\t %d ", ArrayAT);

}

printf("\n");

}

printf("\t\n");

//printf("ЧИСЛО = \n%5d ", num1);

// free(a);

//getchar(); getchar();

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

int\* b; // вказівник на масив

// виділення пам'яті

//malloc - функція для визначення розміру масиву в байтах

// int sizeof() - для точного визначення розміру елементу

// виділення пам'яті

b = (int\*)malloc(n \* m \* sizeof(int));// n·m·(розмір елементу) - об'єм пам'ті необхідний для розміщення двовимірного масиву

//printf("%d Розмір масиву\n\t = ", a);

// ввести кожен елемент масиву

for (i = 0; i < n; i++) // цикл по рядкам

{

for (j = 0; j < m; j++) // цикл по стовпцям

{

printf("b[%d][%d] = ", i, j);//index = i\*m+j;

scanf\_s("%d", (b + i \* m + j)); // а - вказівник на масив, m - кількість стовпців i - індект рядка j індект стовпця

}

}

printf("\nmatrix B\t\n ");

// ввести кожен елемент масиву

for (i = 0; i < n; i++) // по рядкам

{

for (j = 0; j < m; j++) // цикл по стовпцям

{

//\*(a + i \* m + j) звернення до елементу index = i\*m+j; // кожен елмен

printf("%5d ", \*(b + i \* m + j)); // поле шириною 5 символів під елмент масиву

}

printf("\n ");

}

int ArrayB = \*(b + i \* m + j);

//Множення матриць

//(ArrayA) (ArrayB);

int\* c;

c = (int\*)malloc(n \* m \* sizeof(int));

int k;

for (i = 0; i < n; i++)// ідентична частина, але результат виводиться в консоль

{

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

{

for (k = 0; k < m; k++) // кількість стовпців і рядків однакова

{

\*(c + i \* m + j) = 0;

/\*int tmp2 = (\*(b + k \* m + j));

int tmp3 = (\*(a + i \* m + k));\*/

\*(c + i \* m + j) += (\*(a + k \* m + i)) \* (\*(b + k \* m + j));

/\*\*(c + i \* m + j) = (\*(a + i \* m + k )) \* (\*(b + k \* m + j));\*/

//int nk = (\*(a + k \* m + i)) \* (\*(b + k \* m + j));

//printf("\n%d - [%d][%d]", &nk, i,j);

// printf("\n%d - [%d][%d]", \*(c + i \* m + j), i,j);

}

}

}

printf("\nmultiplication Matrix Result А \* В( A transpoed )\n");

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

{

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

printf("%5dc ", \*(c + i \* m + j));

printf("\n");

}

/\*int ArrayAxB=\* (c + 1 \* m + 1) = (\*(a + 1 \* m + 1)) \* (\*(b + 1 \* m + 1));

printf("%5dC ", ArrayAxB);\*/

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

// Додавання матриць

int\* d;

d = (int\*)malloc(n \* m \* sizeof(int));

printf("\n sum of matrics А + В = \n");

for (i = 0; i < n; i++)// ідентична частина, але результат виводиться в консоль

{

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

{

\*(d + i \* m + j) = \*(a + i \* m + j) + \*(b + i \* m + j);

printf("\t %5d ", \*(d + i \* m + j));

}

printf("\n");

}

printf("\t\n");

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// блок пошуку макимального та мінімального елеманта матриці А

int MinNum, MaxNum;

MinNum = MaxNum = \*(a + 0 \* m + 0);

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

{

// проходимо кожний стовпчик строки і

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

{

//перевіряємо кожен елемент масива з максимумом

if (\*(a + j \* m + i) > MaxNum)

{

MaxNum = \*(a + j \* m + i);

}

if (\*(a + j \* m + i) < MinNum)

{

MinNum = \*(a + j \* m + i);

}

}

}

printf("\t\nThe maximum element of the matrix А - %d\n", MaxNum); //вивести максимальний елемент

printf("\t\n smallest element of the matrix А - %d\n", MinNum); //вивести найменший елемент

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//блок сортування матриць за зростанням

int sort;

printf("\n\tMatrix sorting А za zrostaniam ");

sort = \*(a + j \* m + i);

for (int k = 0; k < n \* m; ++k) {

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

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

if (j != n - 1) {

if (\*(a + (j + 1) \* m + i) < \*(a + j \* m + i)) {

int tmp = \*(a + (j + 1) \* m + i);

\*(a + (j + 1) \* m + i) = \*(a + j \* m + i);

\*(a + j \* m + i) = tmp;

}

}

else {

if ((\*(a + 0 \* m + (i + 1)) < \*(a + j \* m + i)) && (i != n - 1)) {

int tmp = \*(a + 0 \* m + (i + 1));

\*(a + 0 \* m + (i + 1)) = \*(a + j \* m + i);

\*(a + j \* m + i) = tmp;

}

}

}

}

}

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

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

printf("\t%d", \*(a + j \* m + i));

}

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

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int z[size] = {24, 12, 4, 1, 30, 12, 23, 4}; // всі значення які задаються для кожного елементу масиву

int max, min, Max2, index; // Max2 зберігає максимальне значення елементу масива і index індекс цього елементу

printArray(z, size); // за допомогою функцій printArray, виконується виклик функції z

max = search\_Max(z, size);

printf("\n\nmax=%d", max); //вивід зміної max

min = search\_Min(z, size);

printf("\n\nmin=%d", min);//вивід зміної min

search\_Max2(z, size, &Max2, &index);

printf("\n\nMax2=%d index=%d \n\n", Max2, index);

sort\_1D(z, size); // для сортування масиву

printf("\n\n\nSorted array:\n"); //виведення відсортованого масиву

printArray(z, size);

printf("\n");

//printArray2( z, size); // z адреса першого елемента масиву, зберігаэться в імені масиву, size розмір масиву

//printf("\n");

//printArray3(z, size); // z адреса першого елемента масиву, зберігаэться в імені масиву, size розмір масиву

//printf("\n");

//int y[size] = { 1, 8, 12, 15 }; // всі елементи після 4 масиву будуть автоматично обнулятися

//int r[size] = { 0 }; //обнулення всіх елементів масиву

//int t[size] = { 4 };// лише перший елемент масиву отримає значення 4, всі інші будуть = 0

return 0;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

void printArray(int \*Z, int S ) // S локальна змінна яка отримує значення size

{

int j;

for (j = 0; j <= S - 1; j++) { // контроль щоб не виходити за межі масиву

//for (j = S-1; j >= 0; j--)

printf("%4d", Z[j] );

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int search\_Min( int \*z, int S)

{

int temp\_min = z[0]; //змінна temp\_max отримує значення першого елеметну масиву

int j;

for (j = 1; j <= S - 1; j++) { // контроль щоб j не виходив за межі масиву, максимальне значення для j це S-1

if (z[j] < temp\_min) { //перевірка якщо на черговий елемент масиву z через індекс j

temp\_min = z[j];

printf("\ntemp\_max=%d", temp\_min); // вивід значення зміної temp\_max для кожного значення

//getch();

}

}

return temp\_min; //повернутися в точку виклику search\_Max(z, size);

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int search\_Max(int \*z, int S)

{

int temp\_max = z[0]; //змінна temp\_max отримує значення першого елеметну масиву

int j;

for (j = 1; j <= S - 1; j++) { // контроль щоб j не виходив за межі масиву, максимальне значення для j це S-1

if (z[j] > temp\_max) { //перевірка якщо на черговий елемент масиву z через індекс j

temp\_max = z[j];

printf("\ntemp\_max=%d", temp\_max); // вивід значення зміної temp\_max для кожного значення

//getch();

}

}

return temp\_max; //повернутися в точку виклику search\_Max(z, size);

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

void search\_Max2( int \*z, int S, int \* pMax, int \* pIndex) // pMax- зберігає адресу зміінної Max2, pIndex- зберігає адресу зміної index

{

int temp\_max;

int temp\_index;

int j;

temp\_max = z[0];

temp\_index = 0;

for (j = 1; j <= S - 1; j++) {

if (z[j] > temp\_max) {

temp\_max = z[j];

temp\_index = j;

}

}

\*pMax = temp\_max; // вміст змінної Max2

\*pIndex = temp\_index; //вміст зміної index

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

void sort\_1D(int \*z, int S) // сортування масиву за зростанням

{

int pass;

int j;

int temp;

for (pass = 1; pass <= S - 1; pass++) { // змінна pass пройде значення 1, 2, 3

for (j = 0; j <= S - 1; j++) {

if (z[j] < z[j + 1]) { // елемент j буде більший за сусідній елемент j+1, тоді в цьому випадку міняємо їх місцями

temp = z[j];

z[j] = z[j + 1];

z[j + 1] = temp;

}

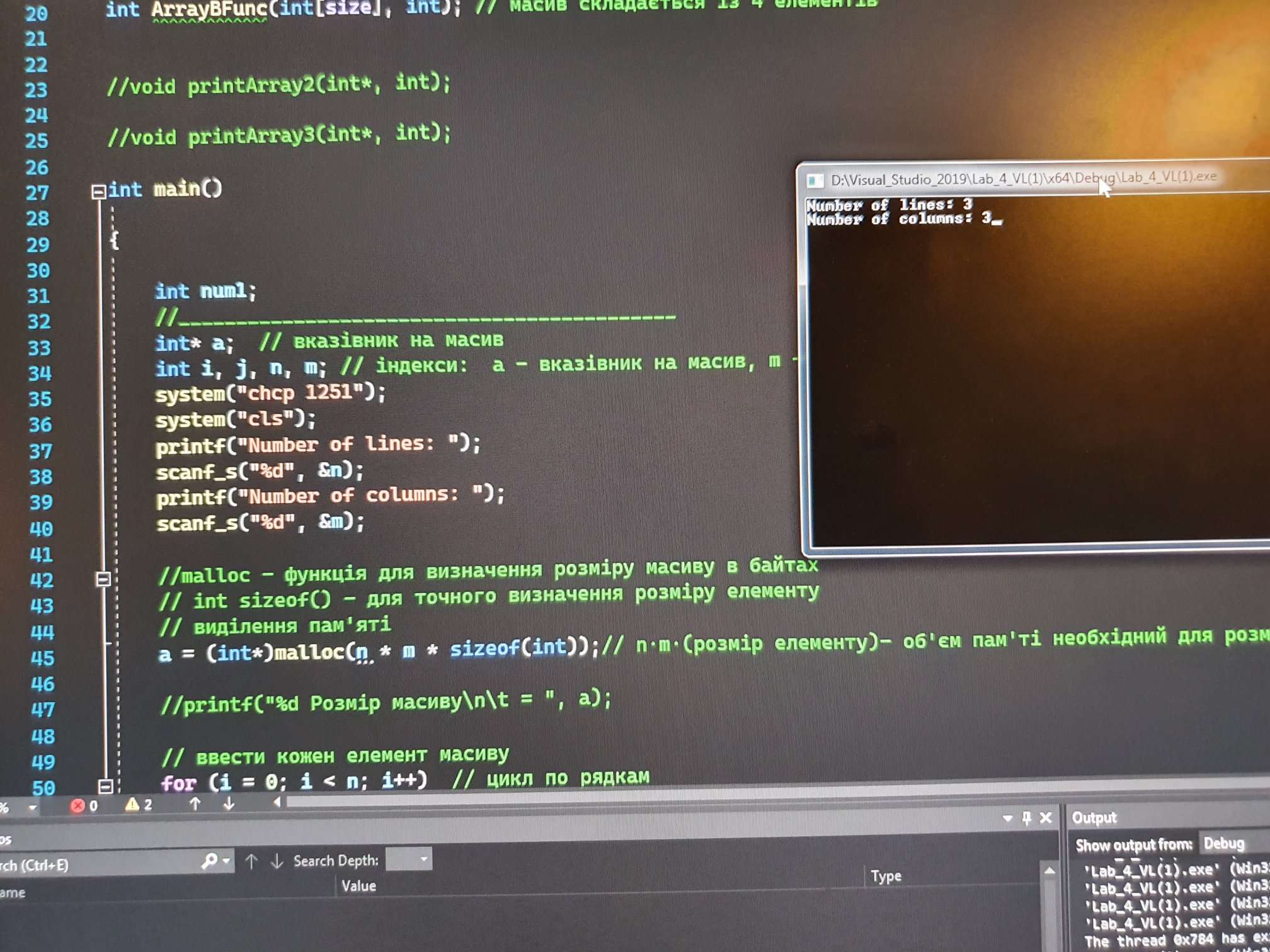
}

}

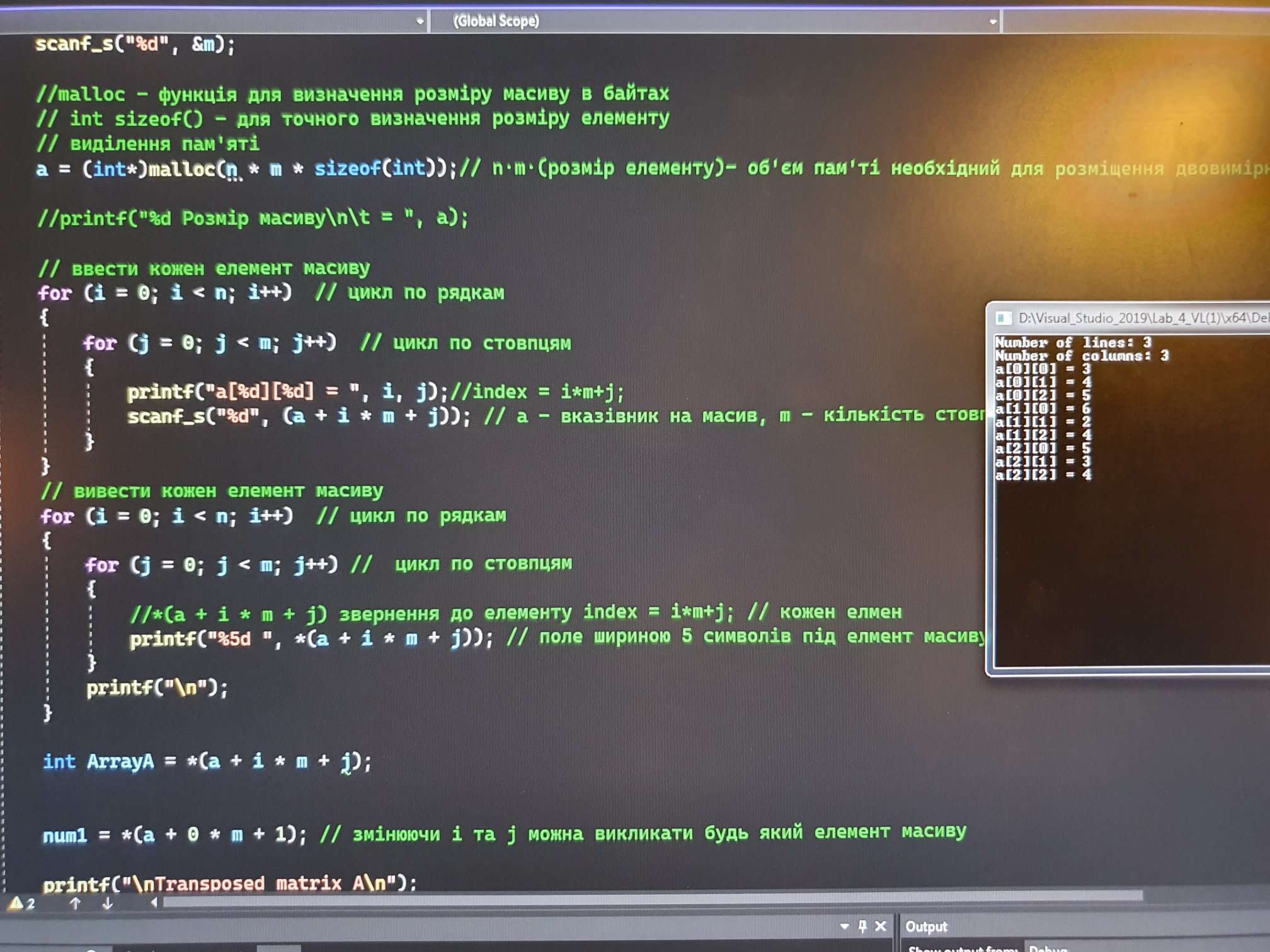
}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

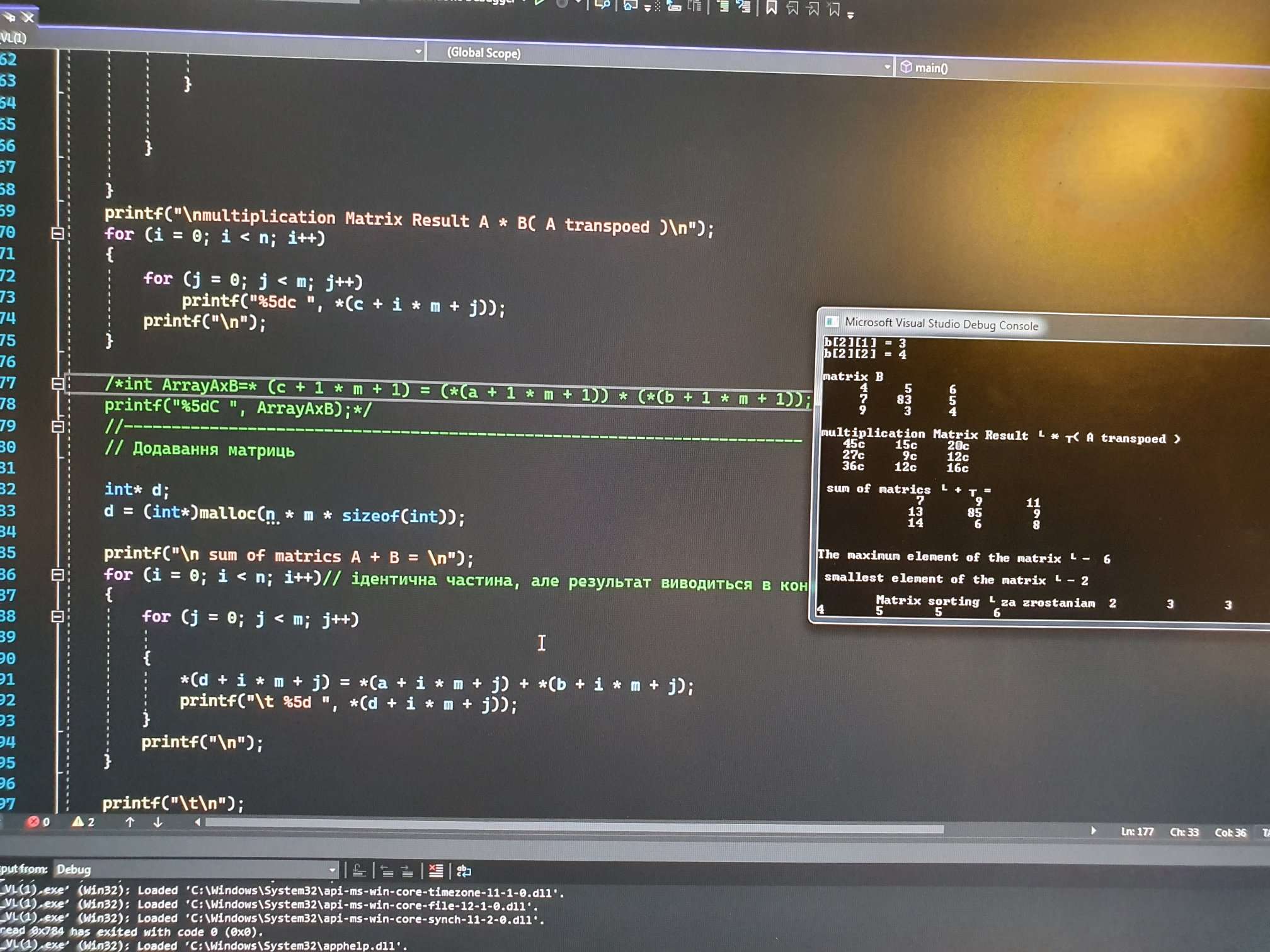
1. Введення кількості рядків і стовпців



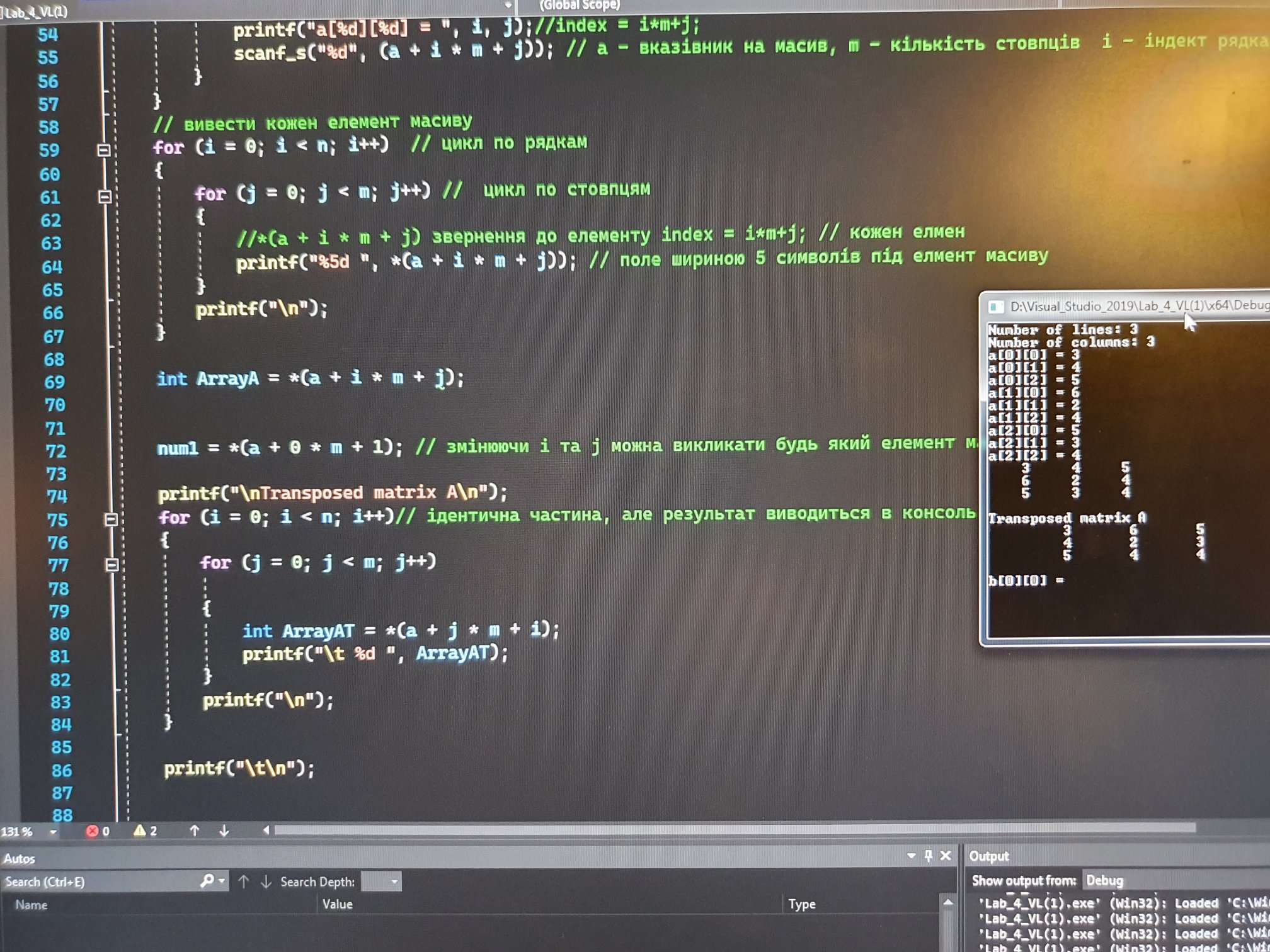
1. Виконання циклу побудови матриці А



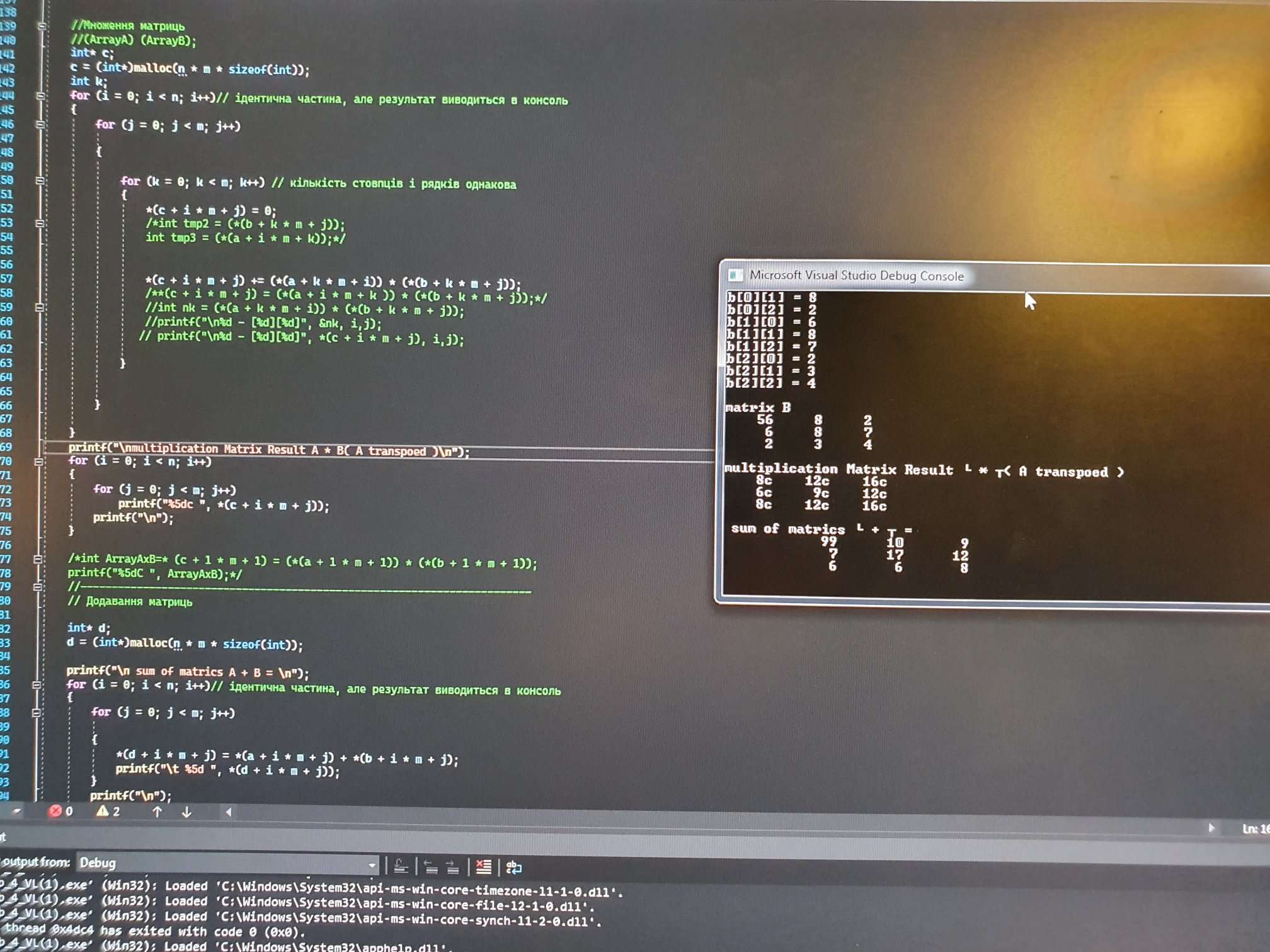
1. Транспонування матриці А



1. Виконання циклу побудови матриці В



1. Множення і додавання матриці



1. Сортування матриці за зростанням

