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 Віктор Андрійович

Група: РЕ-12

Викладачі: доцент Катін П.Ю.

Оцінка: _____

Підпис: _____

Київ – 2021

Мета: скласти програми для роботи з двовимірними масивами.

Код

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <time.h>
#include < malloc.h >  // для використання функцій динамічного розподілу пам'яті


const size_t SIZE = 30;
void CreateValues(int* iArr, size_t size);
void RecursionSort(int* iArr, size_t l, size_t r);


int main()
{
	int num1;


	//_____________________________________________
```

```
	int* a;  // вказівник на масив
	int i, j, n, m;
	system("chcp 1251");
	system("cls");
	printf("Кількість рядків: ");
	scanf_s("%d", &n);
	printf("Кількість стовпців: ");
	scanf_s("%d", &m);
	// виділення пам'яті
	//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++)  // циклпо 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)); // a - вказівник на масив, m - кількість стовпців  i - індект рядка  j індект стовпця
		}
	}
```

```
	// вивести 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 ", *(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 i та j можна викликати будь який
елемекнт масиву

	printf("\nТранспонована матриця 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");
int* b;
b = (int*)malloc(n * m * sizeof(int));

for (i = 0; i < n; i++)
        {
                for (j = 0; j < m; j++)
                {
                        printf("b[%d][%d] = ", i, j);
                        scanf_s("%d", (b + i * m + j));
                }
        }
printf("\nМатриця B\t\n ");

        for (i = 0; i < n; i++)
        {
                for (j = 0; j < m; j++)
                {

                        printf("%5d ", *(b + i * m + j));
                }
                printf("\n ");
```

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

int* c;
        c = (int*)malloc(n * m * sizeof(int));
        int k;
        for (i = 0; i < n; i++)
        {
                for (j = 0; j < m; j++)

                {

                        for (k = 0; k < m; k++)
                        {
                                *(c + i * m + j) = 0;


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

                        }


                }

        }
```

```
printf("\nРезультат множення матриць A * B( A транспонована)\n");
	for (i = 0; i < n; i++)
	{
		for (j = 0; j < m; j++)
			printf("%5dc ", *(c + i * m + j));
		printf("\n");
	}

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

	printf("\nСума матриць A + B = \n");
	for (i = 0; i < n; i++)
	{
		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");
int MinNum, MaxNum;
```

```
MinNum = MaxNum = *(a + 0 * m + 0);
for (i = 0; i < n; i++)
{
	// проходимо кожний стовпчик строки i
	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);



		}

	}
}
//вивести максимальний елемент
printf("\t\nМаксимальний елемент матриці A -  %d\n", MaxNum);
```

```
	//вивести найменший елемент
	printf("\t\nНайменший елемент матриці A - %d\n", MinNum);


int sort;
	//блок сортування матриць за зростанням ______
	printf("\n\tСортування матриці A за зростанням ");
	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");
srand(time(0));


        printf("\t\n\n");
int* iArr = (int*)calloc(SIZE, sizeof(int));
        CreateValues(iArr, SIZE);
printf("RecursionSort\t");
        RecursionSort(iArr, 0, SIZE - 1);
        CreateValues(iArr, SIZE);
        system("pause");
        system("pause");


        return 0;
}
void CreateValues(int* iArr, size_t size) //
{
```

```
	for (size_t i = 0; i < SIZE; ++i) // функція задає випадкові значення в
діапазоні : 128 - 64
		iArr[i] = rand() % 128 - 64;
}
void RecursionSort(int* iArr, size_t l, size_t r)
{
	int pivot = iArr[(l + r) / 2]; // роздільна здатність
	size_t lMargin = l;//ліва границя
	size_t rMargin = r;// права границя
	while (l<r)//// поки границі не зімкнуться
	{
		while ((iArr[r] >= pivot) && (l<r))
			--r;// зсуваємо праву границю поки елемент [right] більше
[pivot]
		if (l != r)// якщо границі не зімкнулися
		{
			iArr[l] = iArr[r];// Переміщуємо елемент [right]
			++l;
		}

		while ((iArr[l] <= pivot) && (l < r))
			++l;// зсуваємо ліву границю, поки елемент [left] менше
[pivot]
		if (l != r) //якщо границі не зімкнулись
		{
			iArr[r] = iArr[l];; // Переміщуємо елемент [left] на місце
[right]
			--r;// Зсув правого кордону вправо
```

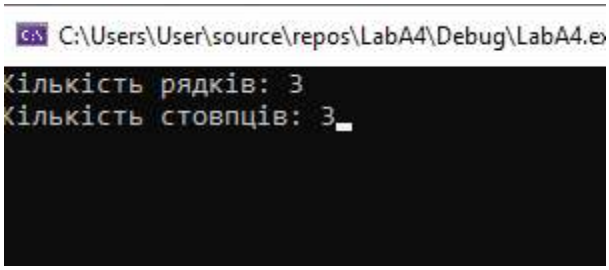
```
                }


        }
        iArr[l] = pivot; // Ставимо дозвільний елемент на місце
        pivot = l;
                                                        //printf("%d\t%d\t", l,r);
                                                        //printf("%d\t%d\t",
iArr[l], iArr[r]);
//printf("%d\t", pivot);//
        l = lMargin;
        r = rMargin;


        if (l < pivot) // Рекурсивно викликаємо сортування для лівої та правої
частини масиву
                RecursionSort(iArr, l, pivot - 1);
        if (r > pivot)
                RecursionSort(iArr, pivot + 1, r);
}
```

Висновок.

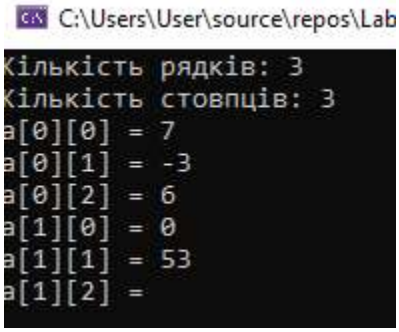
1) Мета роботи - навчитись працювати з двовимірними динамічними масивами.

2)

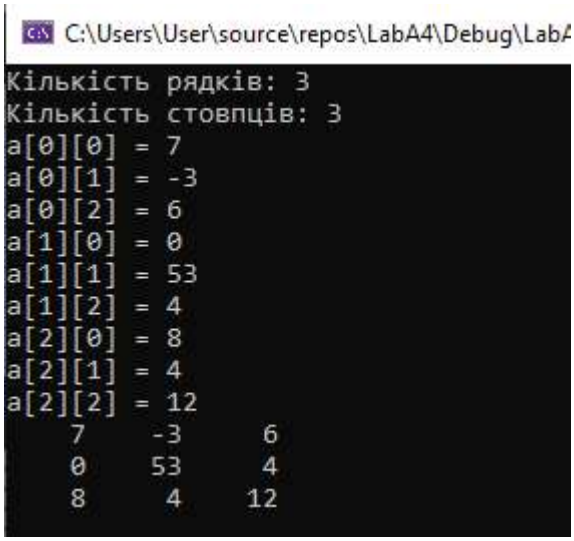
Ввести розмірність матриці.



В залежності від заданої кількості рядків та стовпців, кожному елементу матриці присвоюється значення.



Потім ця матриця виводиться у консоль.



І транспонується.

```
Транспонована матриця A
        7       0       8
        -3      53      4
        6       4       12
```

Далі вводимо матрицю B.

```
C:\Users\User\source\repos\LabA4\Debu
Кількість рядків: 3
Кількість стовпців: 3
a[0][0] = 7
a[0][1] = -3
a[0][2] = 6
a[1][0] = 0
a[1][1] = 53
a[1][2] = 4
a[2][0] = 8
a[2][1] = 4
a[2][2] = 12
    7    -3     6
    0    53     4
    8     4    12

Транспонована матриця A
        7       0       8
        -3      53      4
        6       4       12

b[0][0] = 5
b[0][1] = 8
b[0][2] = 43
b[1][0] = -32
b[1][1] = 4
b[1][2] =
```

Вивести матрицю B.

```
b[0][0] = 5
b[0][1] = 8
b[0][2] = 43
b[1][0] = -32
b[1][1] = 4
b[1][2] = 35
b[2][0] = 12
b[2][1] = 9
b[2][2] = 1

Матриця B
     5     8    43
   -32     4    35
    12     9     1
```

Показати результат їхнього множення.

```
Результат множення матриць A * B( A транспонована)
   96c    72c     8c
   48c    36c     4c
  144c   108c    12c
```

Вивести на екран суму елементів матриць

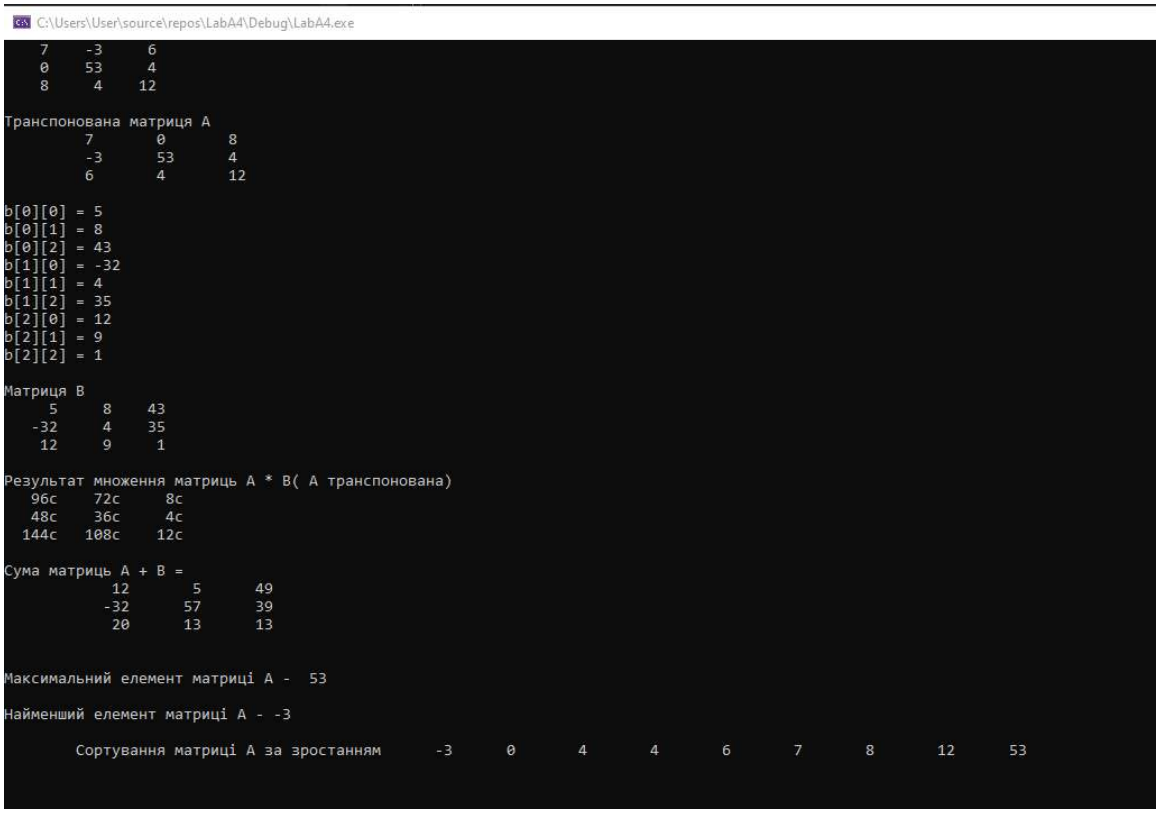
```
Сума матриць A + B =
           12       5      49
          -32      57      39
           20      13      13
```

Вивести на екран найменший і найбільший елемент матриці A.

```
Максимальний елемент матриці A -  53

Найменший елемент матриці A - -3
```

Сортування за зростанням матриці A.

```
      Сортування матриці A за зростанням      -3      0      4      4      6      7      8      12      53
```


C:\Users\User\source\repos\LabA4\Debug\LabA4.exe
7 -3 6
0 53 4
8 4 12
Транспонована матриця A
7 0 8
-3 53 4
6 4 12
b[0][0] = 5
b[0][1] = 8
b[0][2] = 43
b[1][0] = -32
b[1][1] = 4
b[1][2] = 35
b[2][0] = 12
b[2][1] = 9
b[2][2] = 1
Матриця B
5 8 43
-32 4 35
12 9 1
Результат множення матриць A * B(A транспонована)
96c 72c 8c
48c 36c 4c
144c 108c 12c
Сума матриць A + B =
12 5 49
-32 57 39
20 13 13
Максимальний елемент матриці A - 53
Найменший елемент матриці A - -3
Сортування матриці A за зростанням -3 0 4 4 6 7 8 12 53