Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 9 з дисципліни «Алгоритми та структури даних-1. Основи алгоритмізації»

««Дослідження алгоритмів обходу масивів»

Варіант №20

Виконав студент ІП-12 Логвиненко Владислав Олексiйович

(шифр, прізвище, ім'я, по батькові)

Перевірив \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( прізвище, ім'я, по батькові)

Київ 2021

**Лабораторна робота 9 Дослідження алгоритмів обходу**

**масивів**

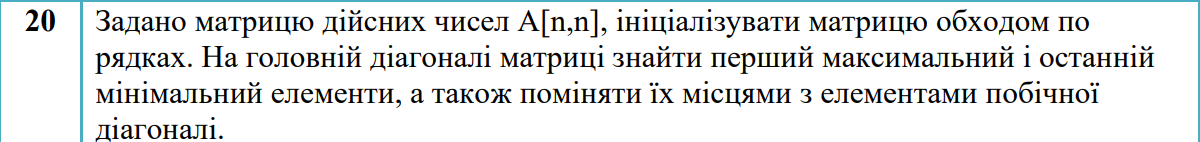
Мета – дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

**Варіант 20**

**Задача.**

Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом.
2. Ініціювання змінної, що описана в п.1 даного завдання.
3. Обчислення змінної, що описана в п.1, згідно з варіантом.



**Постановка задачі**

Знайти значення першого максимального та останнього мінімальних елементів головної діагоналі двовимірного масиву та поміняти їх місцями з відповідними значеннями побічної діагоналі. Ініціалізувати матрицю обходом по рядках. Ініціюємо двовимірний масив випадковими дійсними значеннями в проміжку:[0;10]. Далі знаходимо перший максимальний та останній мінімальний елементи головної діагоналі та їх позиції. Замінюємо ці значення відповідними значеннями побічної діагоналі матриці. Результатом розв’язку буде змінений двовимірний масив.

Побудова математичної моделі.

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Розмір | Ціле | n | Початкове дане |
| Лічильник i | Порядковий | i | Проміжне дане |
| Лічильник j | Порядковий | j | Проміжне дане |
| Функція знаходження макс. елемента та його позиції | Дійсне | max | Проміжні дані |
| Функція знаходження мін. елемента та його позиції | Дійсне | min | Проміжні дані |
| Двовимірний масив | Дійсне | matrix | Проміжні дані,Результат |
| Позиція макс. елемента | Ціле | pos\_1 | Проміжні дані |
| Позиція мін. елемента | Ціле | pos\_2 | Проміжні дані |
| Максисус проміжку | Дійсне | RAND\_MAX | Початкове дане |
| Максимальне значення | Дійсне | а | Початкове дане |

Таким чином математичне формулювання задачі зводиться до створення динамічного двовимірного масиву розмірністю n, потім ініціалізація масиву обходом по рядках(змійкою) випадковими дійсними числами. Знаходження максимального та мінімального елементів та заміна їх на відповідні елементи побічної діагоналі у відповідності до позиції макс. та мін. елементів. Використовую функцію rand() – за допомогою, якої заповняю масив випадковими значеннями.

## 2) Розв’язання

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

*Крок 1.* Визначимо основні дії.

*Крок 2.* Деталізуємо дію введення розміру двовимірного масиву.

*Крок 3.* Деталізуємо дію створення двовимірного масиву.

*Крок 4.* Деталізуємо дію ініціювання масиву обходом по рядках.

*Крок 5.* Деталізуємо дію виведення початкового масиву.

*Крок 6.* Деталізуємо дію знаходження першого максимального елемента та його позиції.

тазнаходження останнього мінімального елемента та його позиції.

*Крок 7.* Деталізуємо дію форматування двовимірного массиву(заміни значень).

*Крок 8.* Деталізуємо дію виведення форматованого масиву.

*Крок 9.* Деталізуємо дію очищення пам’яті.

3) Псевдокод

Крок 1.

**Початок**

введення розміру двовимірного масиву.

створення двовимірного масиву.

ініціювання масиву обходом по рядках.

виведення початкового масиву.

знаходження першого максимального елемента та його позиції.

тазнаходження останнього мінімального елемента та його позиції.

форматування двовимірного массиву(заміни значень).

виведення форматованого масиву.

очищення пам’яті.

.

**кінець**

Крок 2.

**початок**

n ∈ N

створення двовимірного масиву.

ініціювання масиву обходом по рядках.

виведення початкового масиву.

знаходження першого максимального елемента та його позиції.

тазнаходження останнього мінімального елемента та його позиції.

форматування двовимірного массиву(заміни значень).

виведення форматованого масиву.

очищення пам’яті.

**кінець**

Крок 3.

**Початок**

n ∈ N

matrix = input(n).

ініціювання масиву обходом по рядках.

виведення початкового масиву.

знаходження першого максимального елемента та його позиції.

тазнаходження останнього мінімального елемента та його позиції.

форматування двовимірного массиву(заміни значень).

виведення форматованого масиву.

очищення пам’яті.

**кінець**

Крок 4.

**початок**

n ∈ N

matrix = input(n)

set (matrix,n)

виведення початкового масиву.

знаходження першого максимального елемента та його позиції.

тазнаходження останнього мінімального елемента та його позиції.

форматування двовимірного массиву(заміни значень).

виведення форматованого масиву.

очищення пам’яті.

**кінець**

Крок 5.

**початок**

n ∈ N

matrix = **input**(n)

**set** (matrix,n)

**print**(matrix,n)

знаходження першого максимального елемента та його позиції.

тазнаходження останнього мінімального елемента та його позиції.

форматування двовимірного массиву(заміни значень).

виведення форматованого масиву.

очищення пам’яті.

**кінець**

Крок 6.

**Початок**

n ∈ N

matrix = **input**(n)

**set** (matrix,n)

**print**(matrix,n)

min\_max (matrix,n,min,max,pos\_1,pos\_2)

форматування двовимірного массиву(заміни значень).

виведення форматованого масиву.

очищення пам’яті.

**Кінець**

Крок 7.

**Початок**

n ∈ N

matrix = **input**(n)

**set** (matrix,n)

**print**(matrix,n)

min\_max (matrix,n,min,max,pos\_1,pos\_2)

new\_matrix (matrix,n,pos\_1,pos\_2)

виведення форматованого масиву.

очищення пам’яті.

**Кінець**

Крок 8.

**Початок**

n ∈ N

matrix = **input**(n)

**set** (matrix,n)

**print**(matrix,n)

min\_max (matrix,n,min,max,pos\_1,pos\_2)

new\_matrix (matrix,n,pos\_1,pos\_2)

**print**(matrix,n)

очищення пам’яті.

**Кінець**

Крок 9.

**Початок**

n ∈ N

matrix = **input**(n)

**set** (matrix,n)

**print**(matrix,n)

min\_max (matrix,n,min,max,pos\_1,pos\_2)

new\_matrix (matrix,n,pos\_1,pos\_2)

**print**(matrix,n)

delete\_matrix(matrix,n).

**кінець**

**Функція**

**input(n):**

**\*\***matrix = new float\* [n] **повторити для** і **від** 0 **до** n**:** matrix[n] = new float[n] **все повторити**

**return** matrix

**все функція**

**Функція**

**set(\*\*matrix, n):**

**повторити для** і **від** 0 **до** n**:**

**якщо** (i%2 == 0) **то повторити для** j **від** 0 **до** n з кроком 1**:** matrix[i][j]=float (rand())/float (RAND\_MAX)\*a

**все повторити**

**інакше повторити для** j **від** n-1 **до** 0 з кроком -1**:** matrix[i][j]=float (rand())/float (RAND\_MAX)\*a

**все повторити**

**все якщо**

**все повторити** **Все процедура**

**Функція**

**print(\*\*matrix, n):**

**повторити для** і **від** 0 **до** n**: повторити для** j **від** 0 **до** n**:**

виведення matrix[i][j]

**все повторити**

**все повторити**

**все функція**

**Функція**

min\_max (\*\*matrix,n,min,max,pos\_1,pos\_2)

min = matrix[0][0];

max = matrix[0][0];

повторити для I від 0 до n не включно з кроком 1:

якщо I = j && matrix[I][ j] < min

min = matrix[I][ j]

pos\_2 = I;

все якщо

якщо I = j && matrix[I][ j] >= max

max = matrix[I][ j]

pos\_1 = I;

все якщо

все повторити

**все функція**

Функція

New\_matrix(\*\*matrix,n,pos\_1,pos\_2)

Якщо matrix[pos\_1][ pos\_1] != matrix [pos\_1][n -1 - pos\_1]

То

Swap(matrix[pos\_1][ pos\_1], matrix [pos\_1][n -1 - pos\_1])

Все якщо

Якщо matrix[pos\_2][ pos\_2] != matrix [pos\_2][n -1 - pos\_2]

То

Swap(matrix[pos\_2][ pos\_2], matrix [pos\_2][n -1 - pos\_1])

Все якщо

Все функція

Функція   
delete\_matrix(\*\*matrix,n)

Повторити

Для і від 0 до n не включно за кроком 1

Delete[] matrix[i]

Все повторити

Delete[] matrix

***4) Блок-схема***

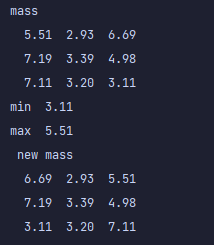
**

***Код програми на мові С++***

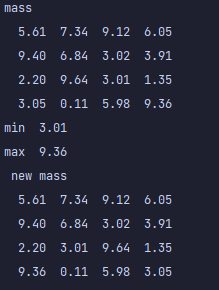
#include <iostream>  
#include <cstdlib>  
#include <ctime>  
*using namespace* std;  
*float* \*\*input(*int* \*n);  
*void* set (*float* \*\*arr\_1,*int* \*n);  
*void* print(*float* \*\*arr\_1,*int* \*n,string massage);  
*void* min\_max(*float* \*\*arr\_1,*int* \*n,*float* \*min,*float* \*max,*int* \*pos\_1,*int* \*pos\_2);  
*void* new\_matrix(*float* \*\*arr\_1,*int* \*n,*int* \*pos\_1,*int* \*pos\_2);  
*void* delete\_matrix(*float* \*\* arr\_1,*int* \*n);  
*int* main(){  
 *int* n,pos\_1,pos\_2;  
 *float* \*\*matrix ,min,max;  
 cout << "Enter size of the array: \n";  
 cin >> n;  
 matrix = input(&n);  
 set(matrix ,&n);  
 print(matrix ,&n,"mass");  
 min\_max(matrix ,&n,&min,&max,&pos\_1,&pos\_2);  
 printf("min %1.2f\n", min);  
 printf("max %1.2f\n", max);  
 new\_matrix(matrix ,&n,&pos\_1,&pos\_2);  
 print(matrix ,&n," new mass");  
 delete\_matrix(matrix ,&n);  
 *return* 0;  
}  
*float* \*\*input(*int* \*n){  
 srand((*unsigned int*)time(**NULL**));  
 *float* \*\*arr\_1 = *new float* \*[\*n];  
 *for* (*int* i = 0;i < \*n;i++){  
 arr\_1[i] = *new float* [\*n];  
 }  
 *return* arr\_1;  
}  
*void* set (*float* \*\*arr\_1,*int* \*n){  
 *for* (*int* i = 0;i < \*n;i++){  
 *if* (i % 2 == 0){  
 *for* (*int* j = 0 ;j < \*n; j++){  
 *float* a = 10.0;  
 arr\_1[i][j] = *float*(rand())/*float*((**RAND\_MAX**)) \* a;  
 }  
 }  
 *else* {  
 *for* (*int* j = \*n - 1;j >= 0; j--){  
 *float* a = 10.0;  
 arr\_1[i][j] = *float*(rand())/*float*((**RAND\_MAX**)) \* a;  
 }  
 }  
 }  
}  
*void* print(*float* \*\*arr\_1,*int* \*n,string massage){  
 cout << massage << endl;  
 *for* (*int* i=0;i<\*n;i++){  
 *for* (*int* j=0;j<\*n;j++){  
 printf(" %1.2f", arr\_1 [i][j]);  
 }  
 cout << endl;  
 }  
}  
*void* min\_max(*float* \*\*arr\_1,*int* \*n,*float* \*min,*float* \*max,*int* \*pos\_1,*int* \*pos\_2){  
 \*min = arr\_1[0][0];  
 \*max = arr\_1[0][0];  
 *for*(*int* i = 0;i < \*n;i++){  
 *for*(*int* j = 0;j < \*n;j++){  
 *if* (i == j && arr\_1[i][j] < \*min){  
 \*min = arr\_1[i][j];  
 \*pos\_2 = i;  
 }  
 *if* (i == j && arr\_1[i][j] >= \*max){  
 \*max = arr\_1[i][j];  
 \*pos\_1 = i;  
 }  
 }  
 }  
}  
*void* new\_matrix(*float* \*\*arr\_1,*int* \*n,*int* \*pos\_1,*int* \*pos\_2){  
 *if* (arr\_1[\*pos\_1][\*pos\_1] != arr\_1[\*pos\_1][\*n - 1 - \*pos\_1 ])  
 swap(arr\_1[\*pos\_1][\*pos\_1],arr\_1[\*pos\_1][\*n - 1 - \*pos\_1 ]);  
 *if* ((arr\_1[\*pos\_2][\*pos\_2] != arr\_1[\*pos\_2][\*n - 1 - \*pos\_2 ]))  
 swap(arr\_1[\*pos\_2][\*pos\_2],arr\_1[\*pos\_2][\*n - 1 - \*pos\_2 ]);  
}  
*void* delete\_matrix(*float* \*\* arr\_1,*int* \*n){  
 *for* (*int* i=0;i<\*n;i++) {  
 *delete*[] arr\_1[i];  
 }  
 *delete*[] arr\_1;  
}

**Віпробування кода:**

**Не парний розмір:**



**Парний розмір:**



**5) Висновок -** Ми дослідили алгоритми обходу масивів та набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій підпрограм. В результаті виконання лабораторної роботи ми отримали алгоритм для знаходження зміненого двовимірного масиву, де перший максимальний та останній мінімальний елементи поміняні з відповідними елементами побічної діагоналі за допомогою підпрограм(функцій та процедур) та з врахуванням умов, декомпозували задачу на 9 кроків: визначили основні дії, знайшли розмір двовимірного масиву, створили його, ініціювали двовимірний масив за допомогою обходу по рядкам, деталізували дію виведення масиву, знайшли перший максимальний та останній мінімальний елемент, деталізували дію форматування масиву(заміни значень), а потім очистили пам’ять.