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 №8 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження алгоритмів пошуку та сортування»

Варіант 18

Виконав студент ІП-12 Кушнір Ганна Вікторівна

(шифр, прізвище, ім'я, по батькові)

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

( прізвище, ім'я, по батькові)

Київ 2021

**Лабораторна робота 8**

**Дослідження алгоритмів пошуку та сортування**

**Мета** – дослідити алгоритми пошуку та сортування, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

**Варіант 18**

*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. Створення нової змінної індексованого типу (одновимірний масив) та її

ініціювання значеннями, що обчислюються згідно з варіантом.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Розмірність** | **Тип даних** | **Обчислення значень елементів одновимірного масиву** |
| **18** | 5×5 | Дійсний | Із від’ємних значень елементів побічної діагоналі двовимірного масиву. Відсортувати методом вставки за спаданням. |

1. *Постановка задачі.* Початковими даними є розмірність двовимірного масиву m×n, за умовою ця розмірність становить 5×5. Оскільки m=n, то доцільно буде використовувати єдину змінну m на позначення кількості рядків та стовпців цієї матриці (масиву). Результатом розв’язку є інша змінна індексованого типу (одновимірний масив), яка складається із від’ємних значень елементів побічної діагоналі першого (двовимірного) масиву та відсортована за спаданням.
2. *Побудова математичної моделі.* Складемо таблицю імен змінних.

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Розмірність двовимірного масиву | Цілий | m | Початкове дане |
| Двовимірний масив | Дійсний | A[m,m] | Допоміжна змінна |
| Розмір одновимірного масиву | Цілий | n | Допоміжна змінна |
| Oдновимірний масив | Дійсний | B[n] | Допоміжна змінна |
| Параметр арифметичного циклу | Цілий, послідовний | i | Лічильник |
| Параметр арифметичного циклу | Цілий, послідовний | j | Лічильник |
| Формальний параметр для передачі розмірності двовимірного масиву у функцію | Цілий | m1 | Допоміжна змінна |
| Формальний параметр для передачі розміру одновим. масиву у функцію | Цілий | n1 | Допоміжна змінна |
| Формальний параметр; двовимірний масив, що передається у функцію | Дійсний | arr2[] | Допоміжна змінна |
| Форм. параметр; одновимірний масив, що передається у функцію | Дійсний | arr1[] | Допоміжна змінна |
| Змінна для перестановки елементів одновимірн. масиву при сортуванні | Дійсний | cop | Допоміжна змінна |

Складемо таблицю імен допоміжних алгоритмів (функцій).

|  |  |  |
| --- | --- | --- |
| **Функція** | **Тип результату** | **Ім’я** |
| Створення одновимірного масиву та визначення кількості його елементів | Цілий | creating\_B() |
| Сортування одновимірного масиву за спаданням методом вставки | – | sorting() |

Таким чином, математичне формулювання задачі зводиться до виконання наступних дій:

1. Ініціалізація змінної m:=5;
2. Введення користувачем двовимірного масиву A[m,m] (розмірністю m на m) або його ініціалізація будь-яким іншим способом;
3. Ініціалізація масиву B[n] від’ємними елементами побічної діагоналі введеного (або згенерованого автоматично) масиву A[m,m] за допомогою функції creating\_B(), а також повернення з цієї функції значення кількості елементів утвореного масиву B[n] за формулою n:=creating\_B(B, A, m).
4. У випадку якщо утворений масив B не містить жодного елемента, виводиться відповідне повідомлення, інакше вібувається виведення отриманого масиву B[n], сортування масиву B[n] за допомогою виклику функції sorting(B, n) та виведення відсортованого масиву.

* creating\_B(arr1[],arr2[],m1) – функція, яка створює одновимірний масив arr1[] з від’ємних елементів побічної діагоналі двовимірного масиву arr2[].

1. На початку ініціалізується змінна n1:=1.
2. Далі за допомогою арифметичного циклу з параметром i (i від 1 до m1 включно) функція виконує лінійний пошук серед елементів побічної діагоналі двовимірного масиву (елементи, які належать до побічної діагоналі: arr2[i,(m1-i+1)]) і у разі, якщо даний елемент від’ємний (arr2[i,(m1-i+1)]<0), значення елемента присвоюється елементу одновимірного масиву з індексом n1 (arr1[n1]) і n1 збільшується на 1 (n1++).
3. Як результат функції повертається вираз (n1-1).

* sorting(arr1[],n1) – функція, яка сортує елементи одновимірного масиву методом вставки за спаданням. Функція складається із арифметичного циклу з параметром i (i від 2 до n1 включно), у якому повторюються наступні дії:

1. Ініціалізуються змінні cop:=arr1[i] та j:=i-1;
2. Далі задається вкладений ітераційний цикл з передумовою (умова: j>=1 && arr1[j]>cop), на кожній ітерації якого виконується зсув відсортованої частини елементів масиву на одну позицію праворуч за формулою: arr1[j+1]:=arr1[j];
3. Виконується вставка елементу a[i] на місце елементу a[j+1] за формулою: a[j+1]:=cop.

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

1. Визначимо основні дії.
2. Деталізуємо ініціалізацію змінної m.
3. Деталізуємо дію створення одновимірного масиву B[] та обчислення кількості його елементів n.
4. Деталізуємо перевірку чи масив B[n] не порожній.
5. Деталізуємо дію сортування масиву B[n] за спаданням.
6. Деталізуємо функцію creating\_B().
7. Деталізуємо функцію sorting().
8. *Псевдокод алгоритму.*

*Крок 1*

**початок**

ініціалізація змінної m

введення A[m,m]

створення одновимірного масиву B[n] і обчислення кількості його елементів

перевірка чи масив B[n] не порожній

**кінець**

*Крок 2*

**початок**

m := 5

введення A[m,m]

створення одновимірного масиву B[n] і обчислення кількості його елементів

перевірка чи масив B[n] не порожній

**кінець**

*Крок 3*

**початок**

m := 5

введення A[m,m]

n := creating\_B(B, A, m)

перевірка чи масив B[n] не порожній

**кінець**

*Крок 4*

**початок**

m := 5

введення A[m,m]

n := creating\_B(B, A, m)

**якщо** (n==0)

**то**

виведення «Побічна діагональ матриці A не містить від’ємних елементів»

**інакше**

виведення B[n]

сортування масиву B[n]

виведення B[n]

**все якщо**

**кінець**

*Крок 5*

**початок**

m := 5

введення A[m,m]

n := creating\_B(B, A, m)

**якщо** (n==0)

**то**

виведення «Побічна діагональ матриці A не містить від’ємних елементів»

**інакше**

виведення B[n]

sorting(B, n)

виведення B[n]

**кінець**

* 1. *Псевдокод допоміжних алгоритмів (функцій).*

*Крок 6*

**початок creating\_B(arr1[],arr2[],m1)**

n1 := 1

**для** i від 1 до m1

**повторити**

**якщо** (arr2[i,(m1 - i + 1)] < 0)

**то**

arr1[n1] := arr2[i,(m1 - i + 1)]

n1++;

**все якщо**

**все повторити**

**повернути** (n1 - 1)

**кінець creating\_B()**

*Крок 7*

**початок sorting(arr1[], n1)**

**для** i від 2 до n1

**повторити**

cpy := arr1[i]

j := i - 1

**поки** (j >= 1 && arr1[j] > cpy)

**повторити**

arr1[j + 1] = arr1[j]

j--

**все повторити**

arr1[j + 1] = cpy

**все повторити**

**кінець sorting()**

1. *Блок-схема алгоритму.*

***Крок 1 Крок 2 Крок 3***

***Крок 4***

***Крок 5***



* 1. *Блок-схеми допоміжних алгоритмів (функцій).*

** *Крок 6*** ***Крок 7***

1. *Код програми (на мові програмування C++).*

#include <iostream>

#include <iomanip>

using namespace std;

void input\_2(float\*\* arr2, int m1)

{

for (int i = 0; i < m1; i++) {

for (int j = 0; j < m1; j++) {

cin >> arr2[i][j];

}

}

cout << endl;

}

void output\_2(float\*\* arr2, int m1)

{

for (int i = 0; i < m1; i++) {

for (int j = 0; j < m1; j++) {

printf("%10.2f",arr2[i][j]);

}

cout << endl;

}

}

void output\_1(float arr1[], int n1)

{

for (int i = 0; i < n1; i++) {

printf("%.2f", arr1[i]);

cout << " ";

}

cout << endl;

}

int creating\_B(float arr1[], float\*\* arr2, int m1) {

int n1 = 0;

for (int i = 0; i < m1; i++) {

if (arr2[i][m1 - i - 1] < 0) {

arr1[n1] = arr2[i][m1 - i - 1];

n1++;

}

}

return n1;

}

void sorting(float arr1[], int n1)

{

int i, j;

float cop;

for (i = 1; i < n1; i++) {

cop = arr1[i];

j = i - 1;

while (j >= 0 && arr1[j] > cop) {

arr1[j + 1] = arr1[j];

j--;

}

arr1[j + 1] = cop;

}

}

int main()

{

int m = 5;

int n;

float\*\* A;

float\* B = 0;

A = new float\*[m];

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

A[i] = new float[m];

}

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

B = new float[m];

}

cout << "Enter the matrix A:" << endl;

input\_2(A, m);

cout << "The matrix A:" << endl;

output\_2(A, m);

n = creating\_B(B, A, m);

if (n == 0) {

cout << "The side diagonal does not contain negative elements!" << endl;

}

else {

cout << "The second massive:" << endl;

output\_1(B, n);

sorting(B, n);

cout << "Sorted one-dimensional array:" << endl;

output\_1(B, n);

}

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

delete[] A[i];

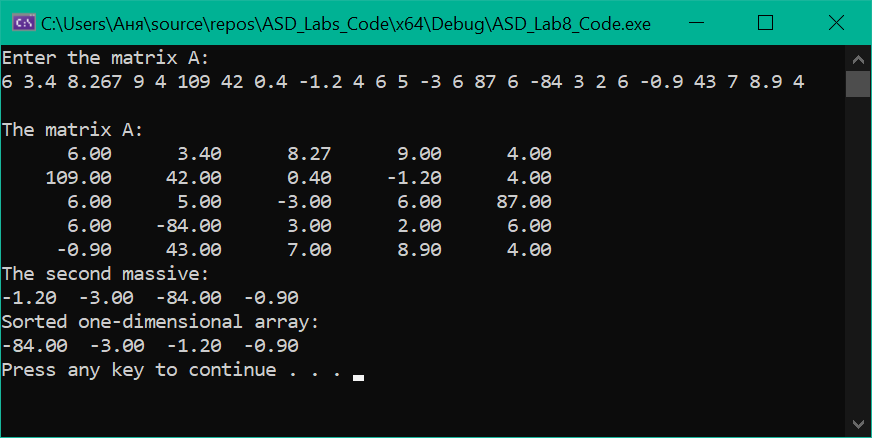
delete[] A;

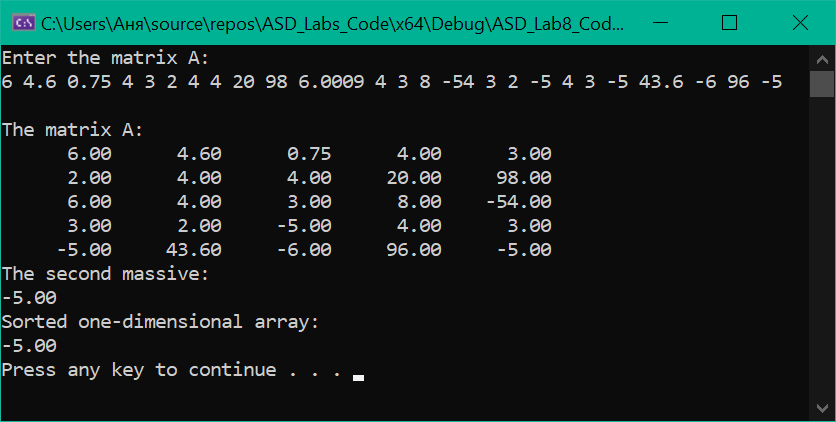
delete[] B;

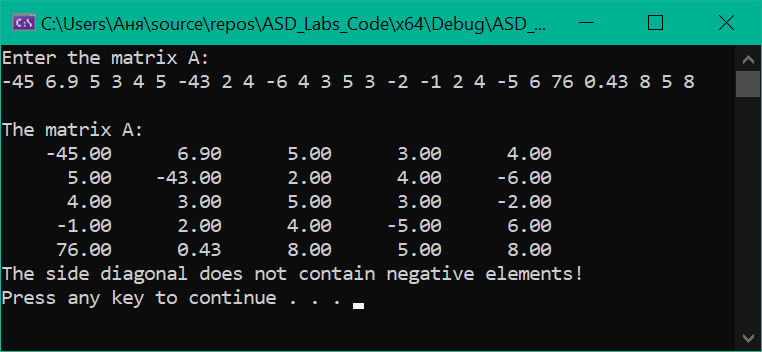
system("pause");

return 0;

}

1. *Тестування програми.*



**

1. *Висновки.* На цій лабораторній роботі було досліджено алгоритми пошуку та сортування, було набуто практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Побудований алгоритм було покладено на мову програмування C++ і написано програму, яку було випробувано з уведенням трьох різних двовимірних масивів.

У першому з уведених масивів побічна діагональ містила 4 від’ємні елементи, які були додані програмою до одновимірного масиву. Цей масив було виведено на екран. Далі було виконано сортування знайденого одновимірного масиву і виведено відсортований масив на екран.

У другому з уведених масивів побічна діагональ містила один від’ємний елемент, який було додано до одновимірного масиву і виведено на екран. Під час сортування ніяких змін з цим масивом не відбулося, тож його знову було виведено на екран як відсортований.

У третьому з уведених масивів побічна діагональ не містила жодного від’ємного елементу, тож на екран було виведено відповідне повідомлення.

Отже, побудований алгоритм працює правильно.