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 3

Виконав студент ІП-15, Борисик Владислав Тарасович
(шифр, прізвище, ім'я, по батькові)

Перевірила Вечерковська Анастасія Сергіївна
(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 3

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.

№ варіанта	Розмірність	Тип даних	Обчислення значень елементів одновимірного масиву
3	6 x 4	Дійсний	Із суми значень елементів рядків двовимірного масиву. Відсортувати методом вставки за зростанням.

Постановка задачі

За умовою задачі потрібно створити два масиви: перший - двовимірний (матриця), другий - одновимірний, ініціювати двовимірний масив дійсними числами, ініціювати одновимірний масив сумами значень елементів рядків двовимірного масиву, відсортувати одновимірний масив методом вставки за зростанням.

Результатом розв'язку є відсортований одновимірний масив.

Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Ім’я	Призначення
Кількість рядків двовимірного масиву	Цілий	rows	Початкове дане
Кількість стовпців двовимірного масиву	Цілий	columns	Початкове дане
Мінімальне значення випадкового числа	Дійсний	min	Проміжне дане

Максимальне значення випадкового числа	Дійсний	max	Проміжне дане
Сума значень елементів рядків двовимірного масиву	Дійсний	sum	Проміжне дане
Проміжне дане для функції sort_array()	Цілий	i	Проміжне дане
Проміжне дане для функції sort_array()	Цілий	j	Проміжне дане
Проміжне дане для функції sort_array()	Дійсний	tmp	Проміжне дане
Двовимірний масив	Дійсний	matrix	Проміжне дане
Одновимірний масив	Дійсний	array	Результат

Для того, щоб показати що аргумент функції передається за посиланням будемо використовувати оператор &.

Для генерації випадкових дійсних чисел будемо використовувати функцію

RandDouble(min, max). Ця функція приймає 2 аргументи:

min - мінімальне значення випадкового числа. Число дійсного типу.
max - максимальне значення випадкового числа. Число дійсного типу.
Функція повертає дійсне число в діапазоні між аргументами min і max включно.

Для ініціювання двовимірного масиву будемо використовувати власну

функцію generate_matrix(&matrix, rows, columns).

Ця функція приймає 3 аргументи:

matrix - матриця (двовимірний масив), яка буде використана для ініціювання.
rows - кількість рядків двовимірного масиву. Число цілого типу.
columns - кількість стовпців двовимірного масиву. Число цілого типу.
Функція за допомогою іншої функції RandDouble(min, max) генерує дійсні випадкові числа і записує їх в двовимірний масив matrix. Нічого не повертає.

Для ініціювання одновимірного масиву сумами значень елементів рядків двовимірного масиву будемо використовувати власну функцію generate_array(matrix, rows, columns, &array).
Ця функція приймає 4 аргументи:

matrix - матриця (двовимірний масив).

rows - кількість рядків двовимірного масиву. Число цілого типу.

columns - кількість стовпців двовимірного масиву. Число цілого типу.
array - одновимірний масив, який буде заповненим сумами значень елементів

рядків двовимірного масиву.
Функція рахує суми значень елементів рядків двовимірного масиву і заповнює ними одновимірний масив array. Нічого не повертає.

Для сортування одновимірного масиву методом вставки за зростанням будемо використовувати власну функцію sort_array(&array, size).
Ця функція приймає 2 аргументи:
array - одновимірний масив, який буде відсортованим.
size - кількість елементів масиву array.
Функція сортує масив array методом вставки за зростанням. Нічого не повертає.

1) Створюємо змінні rows і columns і присвоюємо їм значення 6 і 4 відповідно.
2) Створюємо двовимірний масив matrix.
3) Ініціюємо двовимірний масив matrix за допомогою функції generate_matrix(&matrix, rows, columns).
4) Створюємо одновимірний масив array.
5) Ініціюємо одновимірний масив array за допомогою функції generate_array(matrix, rows, columns, &array).
6) Сортуємо одновимірний масив array за допомогою функції sort_array(&array, size).

Розв'язання

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

Крок 1. Визначимо основні дії

Крок 2. Створення змінних rows і columns і присвоєння їм значення 6 і 4 відповідно.

Крок 3. Створення двовимірного масиву matrix.

Крок 4. Ініціалізація двовимірного масиву matrix за допомогою функції generate_matrix(&matrix, rows, columns).

Крок 5. Створення одновимірного масиву array.
Крок 6. Ініціалізація одновимірного масиву array за допомогою функції generate_array(matrix, rows, columns, &array).
Крок 7. Сортування одновимірного масиву array за допомогою функції

sort_array(&array, size).

Псевдокод

Підпрограма generate_matrix(&matrix, rows, columns):
Початок
min := 1.0
max := 10.0
повторити
для i **від** 0 **до** rows
    повторити
    для j **від** 0 до **columns**
        matrix[i][j] := RandDouble(min, max)
    все повторити
все повторити
Кінець

Підпрограма generate_array(matrix, rows, columns, &array):
Початок
повторити
для i **від** 0 **до** rows
    sum := 0
    для j **від** 0 **до** columns
        sum := sum + matrix[i][j]
    все повторити
    array[i] := sum
все повторити
Кінець

Підпрограма sort_array(&array, size):
Початок
i := 0

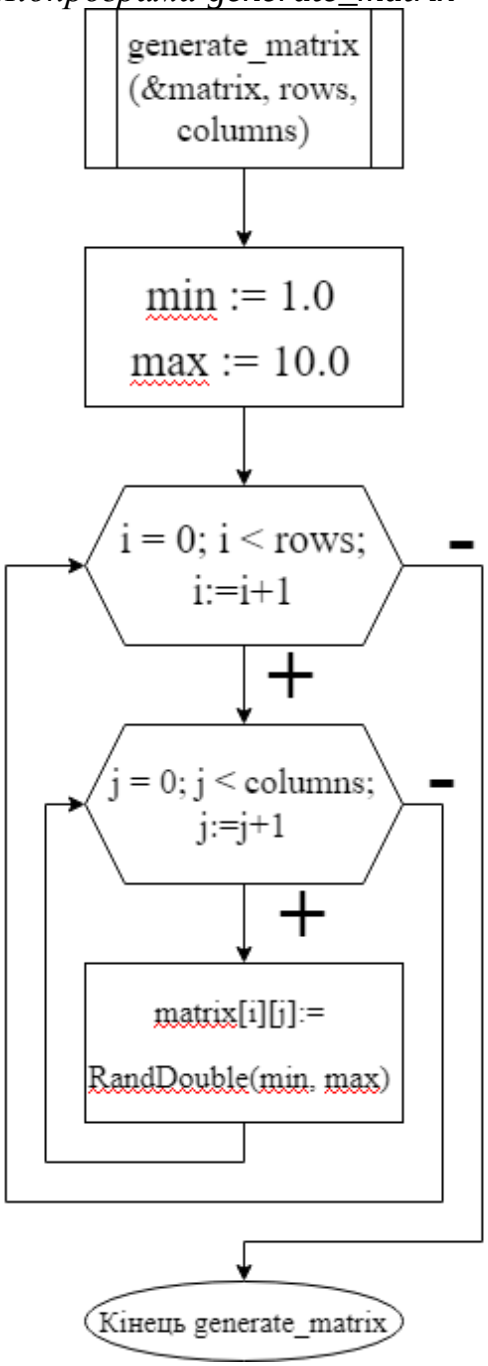
```
j := 0
tmp := 0
повторити
для i від 1 до size
    tmp := array[i]
    для j від i-1 до j >= 0 і array[j] > tmp з кроком -1
        array[j+1] := array[j]
    все повторити
    array[j+1] := tmp
все повторити
Кінець
```

Основна програма:

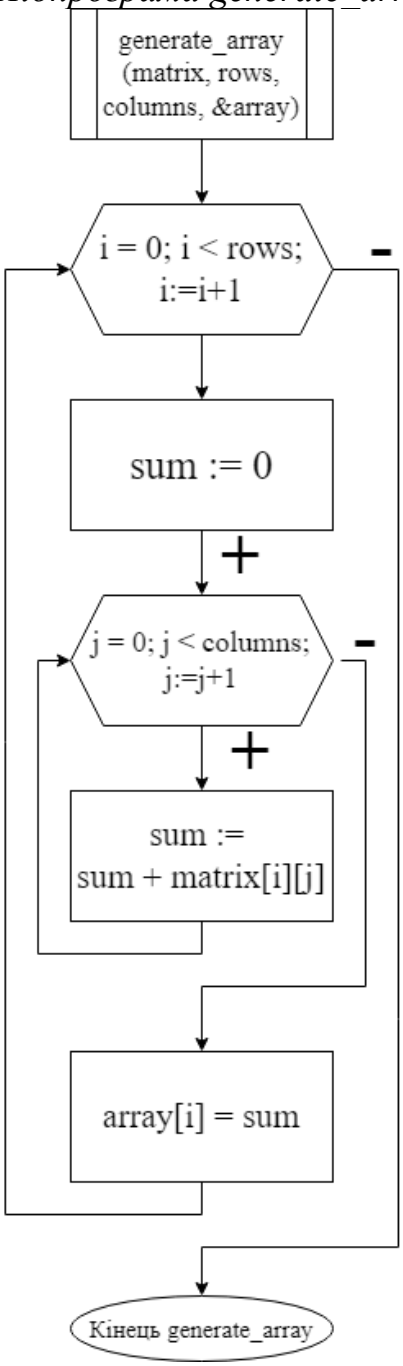
```
rows := 6
columns := 4
matrix[rows][columns]
generate_matrix(matrix, rows, columns)
array[rows]
generate_array(matrix, rows, columns, array)
sort_array(array, rows)
```

Блок-схема алгоритму

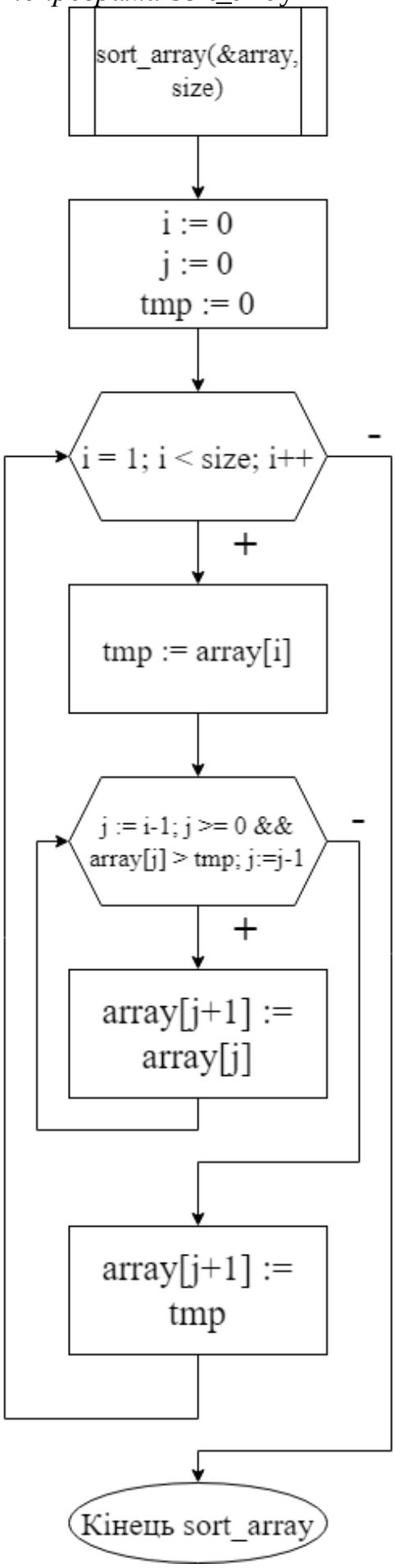
Підпрограма generate_matrix



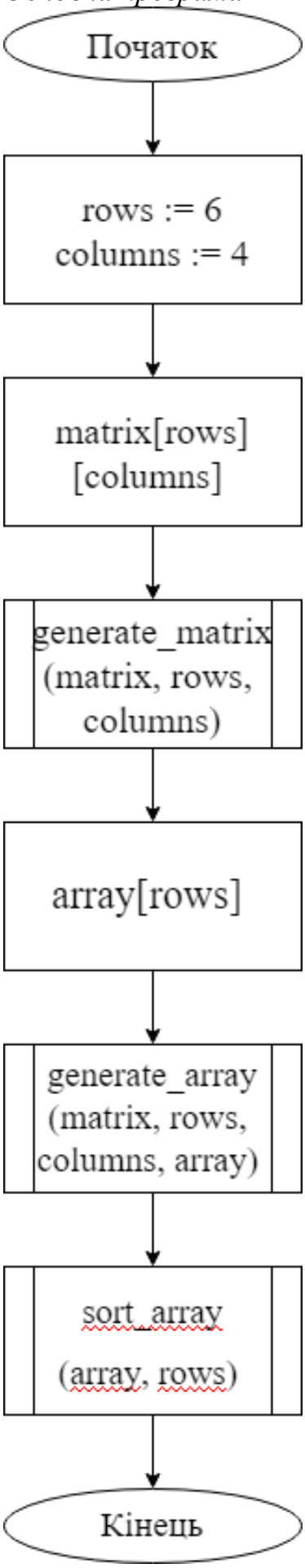
Підпрограма generate_array



Підпрограма sort_array



Основна програма



Код програми

```cpp
#include <iostream>
#include <ctime>
#include <iomanip>
using namespace std;

/** Генерує матрицю заданого розміру
 *  Нічого не повертає
 *  matrix:  матриця (двовимірний масив)
 *  rows:    к-сть рядків
 *  columns: к-сть стовпців  */
void generate_matrix(float** matrix, int rows, int columns);

/** Генерує масив із суми значень елементів рядків двовимірного масиву
 *  Нічого не повертає
 *  matrix:  матриця (двовимірний масив)
 *  rows:    к-сть рядків
 *  columns: к-сть стовпців
 *  array:   масив  */
void generate_array(float** matrix, int rows, int columns, float array[]);

/** Виводить елементи матриці в консоль
 *  Нічого не повертає
 *  matrix:  матриця (двовимірний масив)
 *  rows:    к-сть рядків
 *  columns: к-сть стовпців  */
void print_matrix(float** matrix, int rows, int columns);

/** Виводить елементи масиву в консоль
```

```
 *  Нічого не повертає
 *  array:   масив
 *  rows:    к-сть рядків (к-сть рядків елементів у масиві)  */
void print_array(float* array, int rows);

/** Сортує масив методом вставки за зростанням
 *  Нічого не повертає
 *  array:   масив
 *  size:    к-сть рядків елементів у масиві */
void sort_array(float* array, int size);

int main() {
    srand(time(nullptr));

    //к-сть рядків матриці
    const int rows = 6;
    //к-сть стовпців матриці
    const int columns = 4;

    //створюємо перший двовимірний динамічний масив (матрицю)
    auto **matrix = new float *[rows];
    for(int i = 0; i < rows; i++)
        matrix[i] = new float[columns];

    generate_matrix(matrix, rows, columns);
    print_matrix(matrix, rows, columns);

    float array[rows];
```

```
    generate_array(matrix, rows, columns, array);
    print_array(array, rows);

    sort_array(array, rows);
    cout << "After sorting:\n";
    print_array(array, rows);
}


void generate_matrix(float** matrix, int rows, int columns){
    //заповнюємо матрицю випадковими дійсними числами із діапазону:

    //мінімальне значеннння
    const float min = 1.0;
    //максимальне значеннння
    const float max = 10.0;

    for(int i = 0; i < rows; i++){
        for(int j = 0; j < columns; j++){
            //генеруємо випадкові дійсні числа від 1 до 10:
            //RAND_MAX - константа, значення якої = 32767
            //static_cast<float>(rand()) / RAND_MAX - генерує випадкові дійсні числа від 0 до 1
            //(max - min) * static_cast<float>(rand()) / RAND_MAX - генерує випадкові числа від 0 до 9
            //(max - min) * static_cast<float>(rand()) / RAND_MAX + min - генерує випадкові числа від 1 до 10
            matrix[i][j] = (max - min) * (static_cast<float>(rand()) / RAND_MAX) + min;
        }
    }
}


void generate_array(float** matrix, int rows, int columns, float *array){
    for(int i = 0; i < rows; i++){
        float sum = 0;

        for(int j = 0; j < columns; j++){
            sum += matrix[i][j];
        }

        array[i] = sum;
    }
}


void print_matrix(float** matrix, int rows, int columns){
    cout << "Matrix:\n";

    for(int i = 0; i < rows; i++){
        for(int j = 0; j < columns; j++){
            cout << setprecision( n: 3) << setw( n: 5) << matrix[i][j] << " ";
        }
        cout << "\n";
    }
    cout << "\n";
}
```

```
113 void print_array(float* array, int rows){
114     cout << "Array:\n";
115 
116     for(int i = 0; i < rows; i++)
117         cout << array[i] << " ";
118     cout << "\n\n";
119 }
120 
121 void sort_array(float* array, int size){
122     int i, j;
123     float tmp;
124 
125     //цикл проходу
126     for(i = 1; i < size; i++){
127         tmp = array[i];
128 
129         //шукаємо місце елемента в масиві
130         for(j = i-1; j >= 0 && array[j] > tmp; j--)
131             //сдвигаємо елемент вправо (тому що алгоритм вставки за зростанням)
132             array[j+1] = array[j];
133 
134         //вставляємо елемент
135         array[j+1] = tmp;
136     }
137 }
138 
```

Результат виконання програми

```
"F:\Programs\CLion 2021.1.2\Projects\Labwork_8_ASD\cmake-build-debug\Labwork_8_ASD.exe"
Matrix:
  5.5  2.05  2.15    10
 7.49  8.89  9.47  2.73
 5.41  7.96  9.14  5.81
 2.56  4.29  8.87   4.3
 5.69   6.4  9.29  3.12
 1.94  6.59  5.77  4.86

Array:
19.7 28.6 28.3 20 24.5 19.2

After sorting:
Array:
19.2 19.7 20 24.5 28.3 28.6


Process finished with exit code 0
```

Висновок

Протягом восьмої лабораторної роботи я дослідив алгоритми пошуку та сортування, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій. В результаті виконання роботи я отримав алгоритм, який створює два масиви: перший - двовимірний (матриця), другий - одновимірний, ініціює двовимірний масив дійсними числами, ініціює одновимірний масив сумами значень елементів рядків двовимірного масиву та відсортовує одновимірний масив методом вставки за зростанням.