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 31

Виконав студент ІП-11 Трикош Іван Володимирович

Перевірила Мартинова О. П.

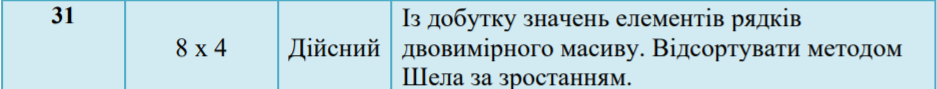
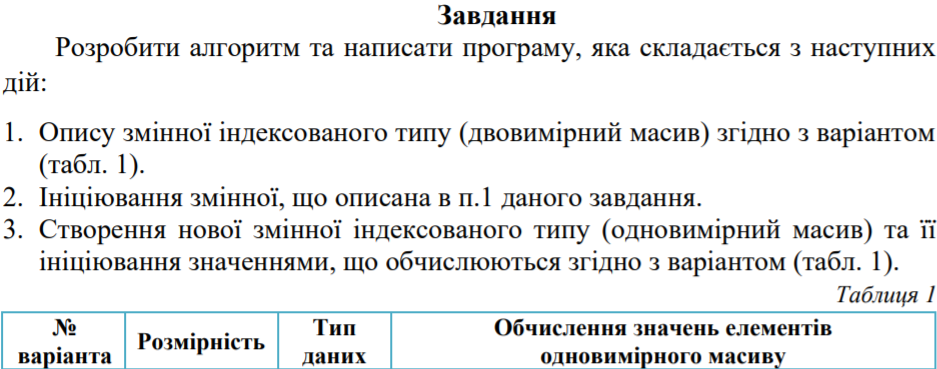
Київ 2021

**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 31.

****

**Постановка задачі –** спочатку потрібно ініціалізувати перший масив та вивести його. Далі знаходити добуток елементів рядка та присвоювати його елементам другого масиву. Потім відсортувати цей масив за зростанням методом Шелла (порівнювати елементи з кроком n/2, n/4, n/8 і т. д.) і вивести його.

**Побудова математичної моделі**

Складемо таблицю імен змінних:

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Кількість рядків | Цілий (const) | rows | Початкове дане |
| Кількість стовпців | Цілий (const) | columns | Початкове дане |
| Перший масив | Дійсний | A | Початкове дане |
| Другий масив | Дійсний | B | Результат |
| Добуток елементів рядка та поточний елемент | Дійсний | b | Проміжне дане |
| Покажчик на масив A | Вказує на дійсний тип | g | Покажчик |
| Покажчик на масив B | Вказує на дійсний тип | p | Покажчик |
| Параметр циклу | Цілий | i | Лічильник |
| Параметр циклу | Цілий | j | Лічильник |
| Параметр циклу | Цілий | k | Лічильник |

Таким чином, математичне формулювання задачі зводиться до ініціалізації першого масиву випадковими дійсними числами, обчислення добутків елементів рядка першого масиву і ініціалізації таким чином другого масиву.

**Псевдокод алгоритму**

*Крок 1. Визначимо основні дії*

*Крок 2. Оголошуємо масиви, їхній розмір та допоміжні змінні*

*Крок 3. Викликаємо та деталізуємо функцію (ініціалізація та виведення першого масиву) і її тіло*

*Крок 4. Викликаємо та деталізуємо функцію (ініціалізація та виведення другого масиву) і її тіло*

*Крок 5. Викликаємо та деталізуємо функцію (обробка другого масиву) і її тіло*

*Крок 6. Викликаємо та деталізуємо функцію (виведення кінцевого масиву) і її тіло*

**Крок 1**

**Початок**

Оголошуємо масиви, їхній розмір та допоміжні змінні

Викликаємо функцію для ініціалізації та виведення першого масиву

Викликаємо функцію для ініціалізації та виведення другого масиву

Викликаємо функцію для обробки другого масиву

Викликаємо функцію для виведення кінцевого масиву

**Кінець**

**Крок 2**

**Початок**

const int rows = 8

const int columns = 4

double A[rows][columns]

double B[rows]

double b = 1.0

Викликаємо функцію для ініціалізації та виведення першого масиву

Викликаємо функцію для ініціалізації та виведення другого масиву

Викликаємо функцію для обробки другого масиву

Викликаємо функцію для виведення кінцевого масиву

**Кінець**

**Крок 3**

**Початок**

const int rows = 8

const int columns = 4

double A[rows][columns]

double B[rows]

double b = 1.0

In\_Out\_A(&A[0][0], rows, columns)

Викликаємо функцію для ініціалізації та виведення другого масиву

Викликаємо функцію для обробки другого масиву

Викликаємо функцію для виведення кінцевого масиву

**Кінець**

**Підпрограма 1**

In\_Out\_A(double\* g, int rows, int columns)

int i = 0

**Повторити** rows **разів**

int j = 0

**Повторити** columns **разів**

\*(g+ i\*columns + j) = (double)(rand())/RAND\_MAX\*10

Вивести \*(g+ i\*columns + j)

j++

**Все повторити**

i++

**Все повторити**

**Кінець**

**Крок 4**

**Початок**

const int rows = 8

const int columns = 4

double A[rows][columns]

double B[rows]

double b = 1.0

In\_Out\_A(&A[0][0], rows, columns)

In\_Out\_B(&A[0][0], B, b, rows, columns)

Викликаємо функцію для обробки другого масиву

Викликаємо функцію для виведення кінцевого масиву

**Кінець**

**Підпрограма 1**

In\_Out\_A(double\* g, int rows, int columns)

int i = 0

**Повторити** rows **разів**

int j = 0

**Повторити** columns **разів**

\*(g+ i\*columns + j) = (double)(rand())/RAND\_MAX\*10

Вивести \*(g+ i\*columns + j)

j++

**Все повторити**

i++

**Все повторити**

**Кінець**

**Підпрограма 2**

In\_Out\_B(double \*g, double\* p, double b, int rows, int columns)

int i = 0

**Повторити** rows **разів**

int j = 0

**Повторити** columns **разів**

b \*= \*(g + i\*columns + j)

j++

**Все повторити**

p[i] = b

Вивести p[i]

b = 1.0

i++

**Все повторити**

**Кінець**

**Крок 5**

**Початок**

const int rows = 8

const int columns = 4

double A[rows][columns]

double B[rows]

double b = 1.0

In\_Out\_A(&A[0][0], rows, columns)

In\_Out\_B(&A[0][0], B, b, rows, columns)

Solution(B, b, rows)

Викликаємо функцію для виведення кінцевого масиву

**Кінець**

**Підпрограма 1**

In\_Out\_A(double\* g, int rows, int columns)

int i = 0

**Повторити** rows **разів**

int j = 0

**Повторити** columns **разів**

\*(g+ i\*columns + j) = (double)(rand())/RAND\_MAX\*10

Вивести \*(g+ i\*columns + j)

j++

**Все повторити**

i++

**Все повторити**

**Кінець**

**Підпрограма 2**

In\_Out\_B(double \*g, double\* p, double b, int rows, int columns)

int i = 0

**Повторити** rows **разів**

int j = 0

**Повторити** columns **разів**

b \*= \*(g + i\*columns + j)

j++

**Все повторити**

p[i] = b

Вивести p[i]

b = 1.0

i++

**Все повторити**

**Кінець**

**Підпрограма 3**

Solution(double\* p, double b, int rows)

int i = rows / 2

**Повторити поки** (i >= 1)

int j = i

**Повторити поки** (j < rows)

int k = j

**Повторити поки** (k >= i)

**Якщо** (p[k - i] > p[k]) **то**

b = p[k]

p[k] = p[k - i]

p[k - i] = b

**Все якщо**

k -= i

**Все повторити**

j++

**Все повторити**

i /= 2

**Все повторити**

**Кінець**

**Крок 6**

**Початок**

const int rows = 8

const int columns = 4

double A[rows][columns]

double B[rows]

double b = 1.0

In\_Out\_A(&A[0][0], rows, columns)

In\_Out\_B(&A[0][0], B, b, rows, columns)

Solution(B, b, rows)

Browse(B, rows)

**Кінець**

**Підпрограма 1**

In\_Out\_A(double\* g, int rows, int columns)

int i = 0

**Повторити** rows **разів**

int j = 0

**Повторити** columns **разів**

\*(g+ i\*columns + j) = (double)(rand())/RAND\_MAX\*10

Вивести \*(g+ i\*columns + j)

j++

**Все повторити**

i++

**Все повторити**

**Кінець**

**Підпрограма 2**

In\_Out\_B(double \*g, double\* p, double b, int rows, int columns)

int i = 0

**Повторити** rows **разів**

int j = 0

**Повторити** columns **разів**

b \*= \*(g + i\*columns + j)

j++

**Все повторити**

p[i] = b

Вивести p[i]

b = 1.0

i++

**Все повторити**

**Кінець**

**Підпрограма 3**

Solution(double\* p, double b, int rows)

int i = rows / 2

**Повторити поки** (i >= 1)

int j = i

**Повторити поки** (j < rows)

int k = j

**Повторити поки** (k >= i)

**Якщо** (p[k - i] > p[k]) **то**

b = p[k]

p[k] = p[k - i]

p[k - i] = b

**Все якщо**

k -= i

**Все повторити**

j++

**Все повторити**

i /= 2

**Все повторити**

**Кінець**

**Підпрограма 4**

Browse(double\* p, int rows)

int і = 0

**Повторити** rows **разів**

Вивід p[i]

і++

**Все повторити**

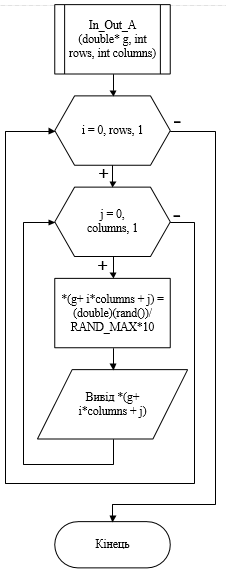
**Кінець**

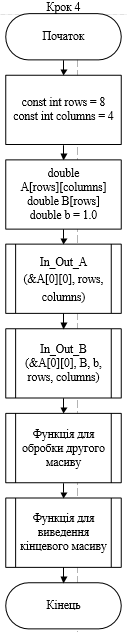
**Блок-схема алгоритму**

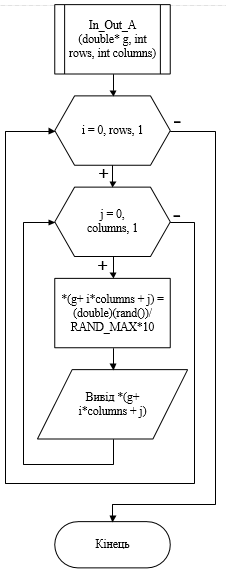
****

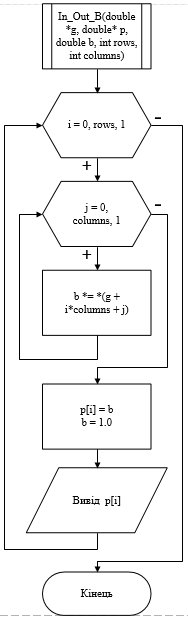
****

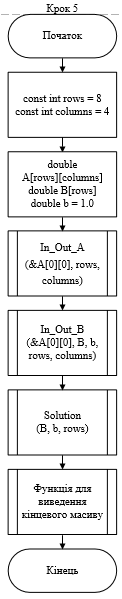
****

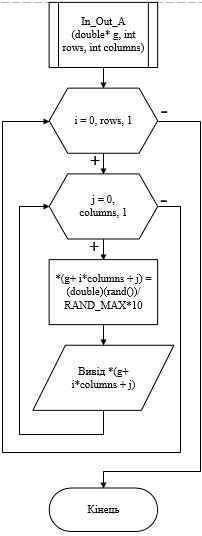
****

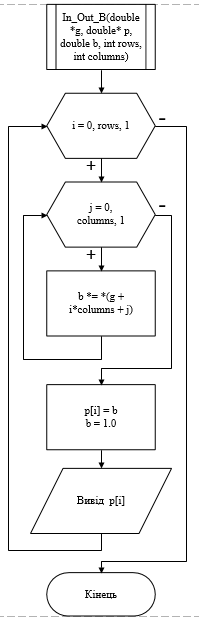
****

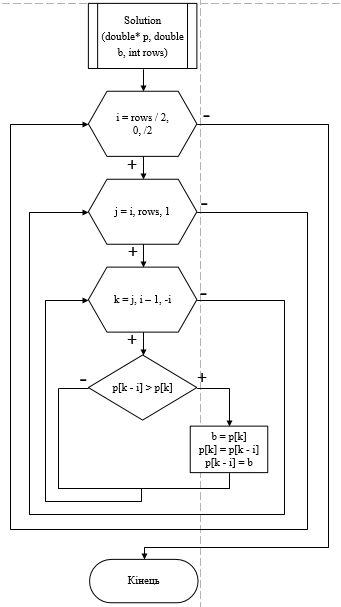
****

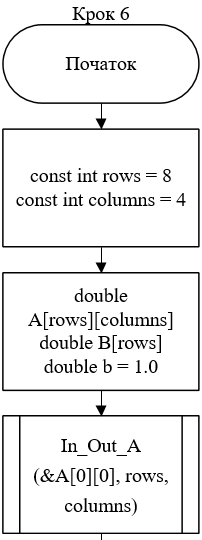
****

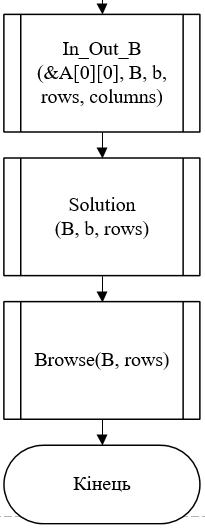
****

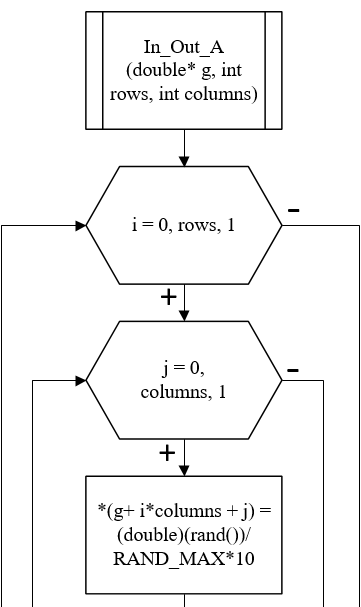
****

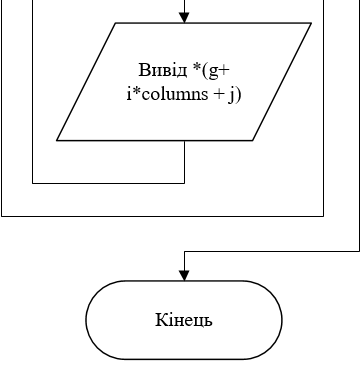
****

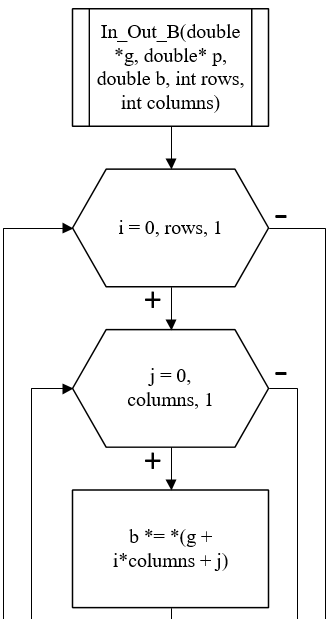
****

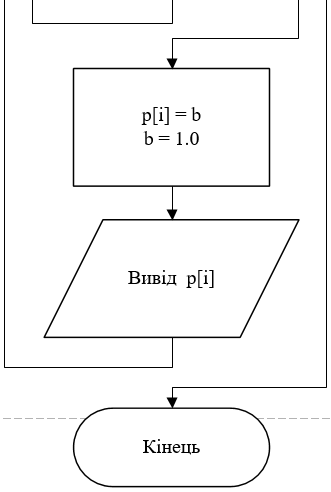
****

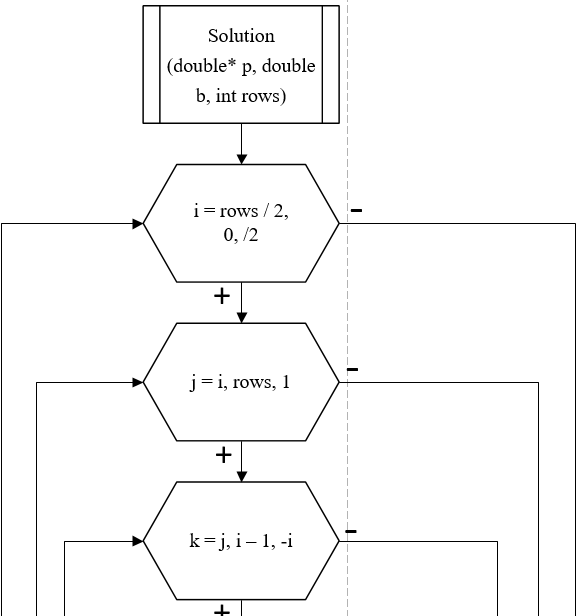
****

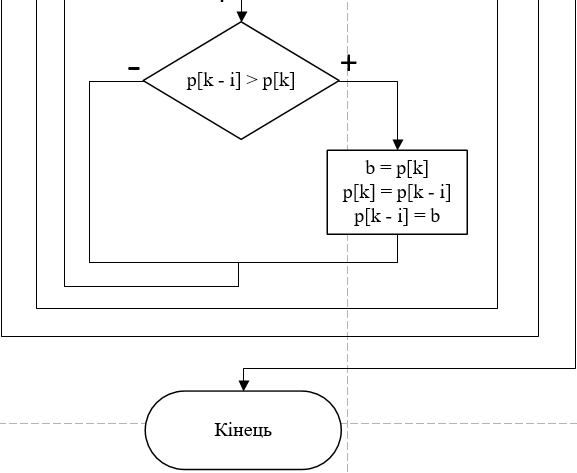
****

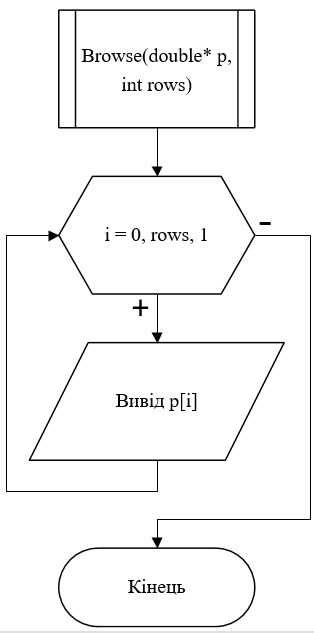
****

****

****

****

****

****

**Код програми на С++**

#include <iostream>

#include <stdlib.h>

#include <iomanip>

using namespace std;

void In\_Out\_A(double\*, int, int); // Ініціалізація та виведення початкового // масиву

void In\_Out\_B(double\*, double\*, double, int, int); // Ініціалізація та виведення // другого масиву

void Solution(double\*, double, int); // Сортування другого масиву

void Browse(double\*, int); // Виведення відсортованого масиву

int main()

{

const int rows = 8; // Кількість рядків

const int columns = 4; // Кількість стовпців

double A[rows][columns]; // Оголошення першого масиву

double B[rows]; // Оголошення другого масиву

double b = 1.0; // Змінна для обчислення добутку елементів рядка та для // заміни елементів

In\_Out\_A(&A[0][0], rows, columns); // Ініціалізація та виведення першого // масиву

In\_Out\_B(&A[0][0], B, b, rows, columns); // Ініціалізація та виведення // другого масиву

Solution(B, b, rows); // Обробка третього масиву

Browse(B, rows); // Виведення обробленого масиву

return 0;

}

void In\_Out\_A(double\* g, int rows, int columns)

{

srand(time(NULL));

cout << "Array A: ";

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

{

cout << "\n";

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

{

\*(g + i \* columns + j) = (double)(rand()) / RAND\_MAX \* 10;//Випадкове // дійсне число

printf(" %.1f", \*(g + i \* columns + j)); // Виводимо поточний елемент // масиву

}

}

}

void In\_Out\_B(double \*g, double\* p, double b, int rows, int columns)

{

cout << "\n\nArray B:\n";

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

{

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

{

b \*= \*(g + i\*columns + j); // Накопичуємо добуток елементів рядка

}

p[i] = b; // Ініціалізуємо елементи другого масиву

printf(" %.4f ", p[i]); // Виводимо поточний елемент масиву

b = 1.0;

}

}

void Solution(double\* p, double b, int rows)

{

for (int i = rows / 2; i >= 1; i /= 2) // Крок обміну

{

// Перебираємо елементи з кроком і

for (int j = i; j < rows; j++)

{

for (int k = j; k >= i; k -= i)

{

// Мінямо елементи місцями, якщо елемент зліва більший за // елемент справа

if (p[k - i] > p[k])

{

b = p[k];

p[k] = p[k - i];

p[k - i] = b;

}

}

}

}

}

void Browse(double\* p, int rows)

{

cout << "\n\nAnswer:\n";

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

{

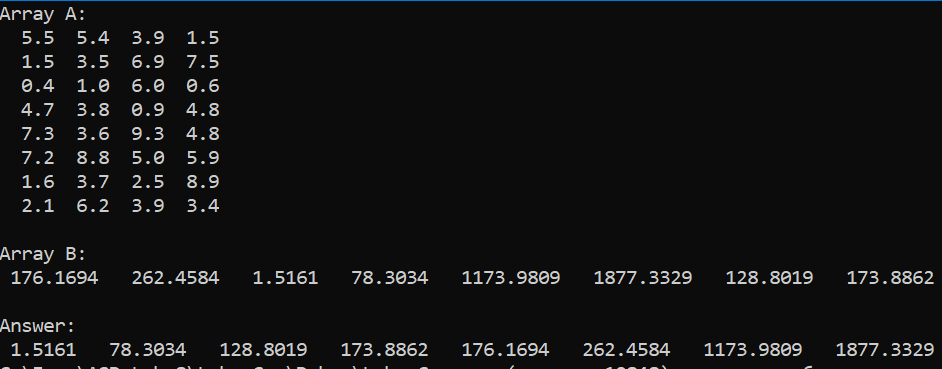
printf(" %.4f ", p[i]); // Виводимо елементи відсортованого масиву

}

}

**Випробування алгоритму**

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Виклик In\_Out\_A(&A[0][0], rows, columns) |
|  |  |
|  | In\_Out\_A(double\* g, int rows, int columns) |
|  | Перша ітерація: |
| 2 | i = 0 |
| 3 | j = 0 |
| 4 | \*(g + i \* columns + j) = 5.5 |
| 5 | Вивід 5.5 |
|  | Остання ітерація: |
| 6 | i = 7 |
| 7 | j = 3 |
| 8 | \*(g + i \* columns + j) = 3.4 |
| 9 | Вивід 3.4 |
|  |  |
| 10 | Виклик In\_Out\_A(&A[0][0], rows, columns) |
|  |  |
|  | In\_Out\_B(double \*g, double\* p, double b, int rows, int columns) |
|  | Перша ітерація: |
| 11 | i = 0 |
| 12 | j = 0 |
| 13 | b = 5.5 |
|  | Остання ітерація: |
| 14 | i = 7 |
| 15 | j = 3 |
| 16 | b = 173.8862 |
| 17 | p[i] = 173.8862 |
| 18 | b = 1.0 |
| 19 | Вивід 173.8862 |
|  |  |
| 20 | Виклик Solution(B, b, rows) |
|  |  |
|  | Solution(double\* p, double b, int rows) |
|  | Перша ітерація: |
| 21 | i = 4 |
| 22 | j = 4 |
| 23 | k = 4 |
| 24 | 176.1694 > 1173 - неправильно |
|  | Остання ітерація: |
| 25 | i = 1 |
| 26 | j = 7 |
| 27 | k = 1 |
| 28 | 1.5161 > 78.3034 - неправильно |
|  |  |
| 29 | Виклик Browse(B, rows) |
|  |  |
|  | Browse(double\* p, int rows) |
| 30 | Вивід 1.5161 78.3034 128.8019 173.8862 176.1694 262.4584 1173.9809 1877.3329 |
|  | Результат 1.5161 78.3034 128.8019 173.8862 176.1694 262.4584 1173.9809 1877.3329 |
|  | Кінець |

****

**Висновок –** я дослідив алгоритми пошуку та сортування, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій, покращив свої навички робити з масивами на мові С++. У програмі я використав алгоритм сортування Шелла за зростанням.

У результаті виконання програми я одержав відсортований за зростанням масив, що складається з добутків елементів рядка двовимірного масиву.