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 27

Виконав студент ІП-11 Савенко Олексій Андрійович
(шифр, прізвище, ім'я, по батькові)

Перевірив Мартинова О. П.
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 8

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

Варіант 27

Індивідуальне завдання

27	8 x 4	Дійсний	Із добутку від'ємних значень елементів рядків двовимірного масиву. Відсортувати обміном за зростанням.

Постановка завдання

Потрібно створити матрицю розмірності 8 x 4 , а після цього одновимірний масив розмірності 8, елементами якого стануть добутки від'ємних значень відповідних рядків двовимірного масиву. Після цього отриманий масив потрібно відсортувати за допомогою обміну за зростанням. Відсортований масив є кінцевим результатом завдання.

Математична модель

Основна программа

main()

Змінна	Тип	Ім'я	Призначення
Кількість рядків матриці	Цілий	rows	Вхідні дані
Кількість стовпців матриці	Цілий	pillars	Вхідні дані
Матриця	Матриця дійсного типу	A	Проміжні дані
Одновимірний масив	Одновимірний масив дійсного типу	B	Результат

Підпрограми

Input()

Змінна	Тип	Ім’я	Призначення
Матриця	Матриця дійсного типу	**Mas**	Проміжні дані, формальний параметр
Кількість рядків матриці	Цілий	**rows**	Проміжні дані, формальний параметр
Кількість стовпців матриці	Цілий	**pillars**	Проміжні дані, формальний параметр
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**i**	Проміжні дані
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**j**	Проміжні дані

Output()

Змінна	Тип	Ім’я	Призначення
Матриця,Одновимірний масив	Матриця дійсного типу, Одновимірний масив дійсного типу	**Mas**	Проміжні дані, формальний параметр
Кількість рядків матриці	Цілий	**rows**	Проміжні дані, формальний параметр
Кількість стовпців матриці	Цілий	**pillars**	Проміжні дані, формальний параметр
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**i**	Проміжні дані
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**j**	Проміжні дані

Create()

Змінна	Тип	Ім’я	Призначення
Матриця	Матриця дійсного типу	**Mas1**	Проміжні дані, формальний

			параметр
Одновимірний масив	Одновимірний масив дійсного типу	**Mas2**	Проміжні дані, формальний параметр
Кількість рядків матриці	Цілий	**rows**	Проміжні дані, формальний параметр
Кількість стовпців матриці	Цілий	**pillars**	Проміжні дані, формальний параметр
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**i**	Проміжні дані
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**j**	Проміжні дані
Добуток від’ємних значень відповідного рядка матриці	Дійсний	**dobutok**	Проміжні дані

Sort()

Змінна	Тип	Ім’я	Призначення
Збереження значень, для обміну елементів	Дійсний	**save**	Проміжні дані
Одновимірний масив	Одновимірний масив дійсного типу	**Mas**	Проміжні дані, формальний параметр, Результат
Кількість рядків матриці	Цілий	**rows**	Проміжні дані, формальний параметр
Кількість стовпців матриці	Цілий	**pillars**	Проміжні дані, формальний параметр
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**i**	Проміжні дані
Лічильник арифметичного циклу задля здійснення операцій з	Цілий	**j**	Проміжні дані

масивом			
Добуток від'ємних значень відповідного рядка матриці	Дійсний	**dobutok**	Проміжні дані

Крок 1. Визначимо та охарактеризуємо основні дії алгоритму

Крок 2. Деталізуємо дію Заповнення матриці

Крок 3. Деталізуємо дію Виведення елементів матриці

Крок 4. Деталізуємо дію Створення одновимірного масиву на основі добутку від'ємних елементів відповідних рядків матриці

Крок 5. Деталізуємо дію Відсортування обміном елементів одновимірного масиву

Крок 6. Деталізуємо дію Виведення відсортованого масиву

Псевдокод алгоритму

Крок 1.

Початок

Деталізуємо дію Заповнення матриці

Деталізуємо дію Виведення елементів матриці

Деталізуємо дію Створення одновимірного масиву на основі добутку від'ємних елементів відповідних рядків матриці

Деталізуємо дію Відсортування обміном елементів одновимірного масиву

Деталізуємо дію Виведення відсортованого масиву

Кінець

Крок 2.

Початок

Input(A, rows, pillars)

Деталізуємо дію Виведення елементів матриці

Деталізуємо дію Створення одновимірного масиву на основі добутку від'ємних елементів відповідних рядків матриці

Деталізуємо дію Відсортування обміном елементів одновимірного масиву

Деталізуємо дію Виведення відсортованого масиву

Кінець

Підпрограми

Input(Mas, rows, pillars)

```
повторити
 для i від 0 до rows із кроком 1
        повторити
                для j від 0 до pillars із кроком 1
                        Mas[i][j] = round(((double)rand()/3200-5)*100)/100
        все повторити
 все повторити
Кінець
```

Крок 3.

Початок

Input(A, rows, pillars)

Output(A, rows, pillars)

Деталізуємо дію Створення одновимірного масиву на основі добутку від'ємних елементів відповідних рядків матриці

Деталізуємо дію Відсортування обміном елементів одновимірного масиву

Деталізуємо дію Виведення відсортованого масиву

Кінець

Підпрограми

Input(Mas, rows, pillars)

```
повторити
 для i від 0 до rows із кроком 1
        повторити
                для j від 0 до pillars із кроком 1
                 Mas[i][j] = round(((double)rand()/3200-5)*100)/100
        все повторити
```

все повторити

Кінець

Output(Mas, rows, pillars)

повторити

для i **від** 0 **до** rows **із кроком** 1

повторити

для j **від** 0 **до** pillars **із кроком** 1

Виведення Mas[i][j]

все повторити

все повторити

Кінець

Крок 4.

Початок

Input(A, rows, pillars)

Output(A, rows, pillars)

Create(A, B, rows,pillars)

Деталізуємо дію Відсортування обміном елементів одновимірного масиву

Деталізуємо дію Виведення відсортованого масиву

Кінець

Підпрограми

Input(Mas, rows, pillars)

повторити
 для i **від** 0 **до** rows **із кроком** 1
 повторити
  для j **від** 0 **до** pillars **із кроком** 1
   Mas[i][j] = round(((double)rand()/3200-5)*100)/100
  все повторити
 все повторити
Кінець

Output(Mas, rows, pillars)

повторити
 для i **від** 0 **до** rows **із кроком** 1
  повторити
   для j **від** 0 **до** pillars **із кроком** 1
    Виведення Mas[i][j]
  все повторити
 все повторити
Кінець

Create(Mas1, Mas2, rows,pillars)

повторити

для i **від** 0 **до** rows **із кроком** 1

dobutok = 1;

повторити

для j **від** 0 **до** pillars **із кроком** 1

якщо

то

dobutok *= Mas1[i][j];

Mas2[i] = round(dobutok*100)/100;

все повторити

все повторити

Кінець

Крок 5.

Початок

Input(A, rows, pillars)

Output(A, rows, pillars)

Create(A, B, rows,pillars)

Sort(B, rows)

Деталізуємо дію Виведення відсортованого масиву

Кінець

Підпрограми

Input(Mas, rows, pillars)

повторити
 для i **від** 0 **до** rows **із кроком** 1
 повторити
  для j **від** 0 **до** pillars **із кроком** 1
   Mas[i][j] = round(((double)rand()/3200-5)*100)/100
  все повторити
 все повторити
Кінець

Output(Mas, rows, pillars)

повторити
 для i **від** 0 **до** rows **із кроком** 1
  повторити
   для j **від** 0 **до** pillars **із кроком** 1
    Виведення Mas[i][j]
  все повторити
 все повторити
Кінець

Create(Mas1, Mas2, rows,pillars)

повторити
 для i **від** 0 **до** rows **із кроком** 1

```
 dobutok = 1;
      повторити
            для j від 0 до pillars із кроком 1
                  якщо Mas1[i][j] < 0
                   то
                      dobutok *= Mas1[i][j];
            Mas2[i] = round(dobutok*100)/100;
      все повторити
 все повторити
Кінець
```

```
Sort( Mas,  rows)

 save;
 повторити
  для i від 0 до rows із кроком 1
      повторити
       для j від 0 до rows - i – 1 із кроком 1
            якщо Mas[j] > Mas[j + 1]
                  то
                        save = Mas[j];
                        Mas[j] = Mas[j + 1];
                        Mas[j + 1] = save;
      все повторити
 все повторити
Кінець
```

Крок 6.

Початок

Input(A, rows, pillars)

Output(A, rows, pillars)

Create(A, B, rows,pillars)

Sort(B, rows)

Output(B, rows)

Кінець

Підпрограми

Input(Mas, rows, pillars)

```
повторити
  для i від 0 до rows із кроком 1
  повторити
        для j від 0 до pillars із кроком 1
          Mas[i][j] = round(((double)rand()/3200-5)*100)/100
        все повторити
  все повторити
Кінець
```

Output(Mas, rows, pillars)

повторити
для i **від** 0 **до** rows **із кроком** 1
повторити
для j **від** 0 **до** pillars **із кроком** 1
Виведення Mas[i][j]
все повторити
все повторити
Кінець

Create(Mas1, Mas2, rows,pillars)

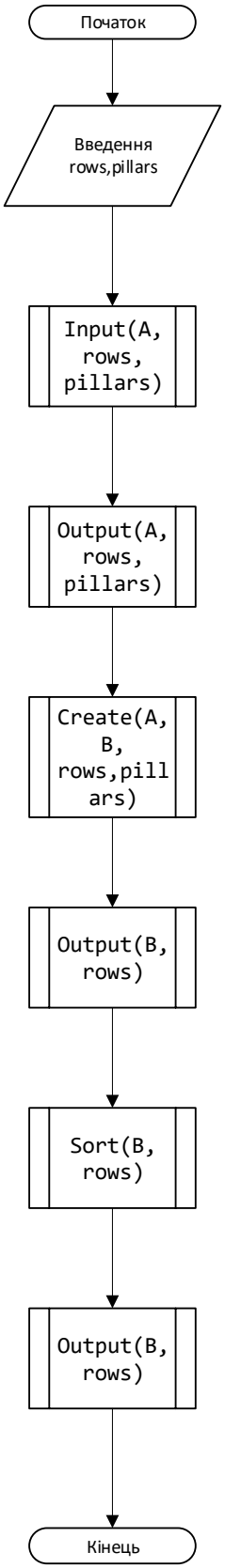
повторити
для i **від** 0 **до** rows **із кроком** 1
dobutok = 1;
повторити
для j **від** 0 **до** pillars **із кроком** 1
якщо Mas1[i][j] < 0
то
dobutok *= Mas1[i][j];
Mas2[i] = round(dobutok*100)/100;
все повторити
все повторити
Кінець

Sort(Mas, rows)

```
 save;
 повторити
  для i від 0 до rows із кроком 1
      повторити
       для j від 0 до rows - i – 1 із кроком 1
            якщо Mas[j] > Mas[j + 1]
                 то
                      save = Mas[j];
                      Mas[j] = Mas[j + 1];
                      Mas[j + 1] = save;
      все повторити
 все повторити
Кінець


Output(Mas, rows)

повторити
 для i від 0 до rows із кроком 1
      Виведення Mas[i]

Кінець
```

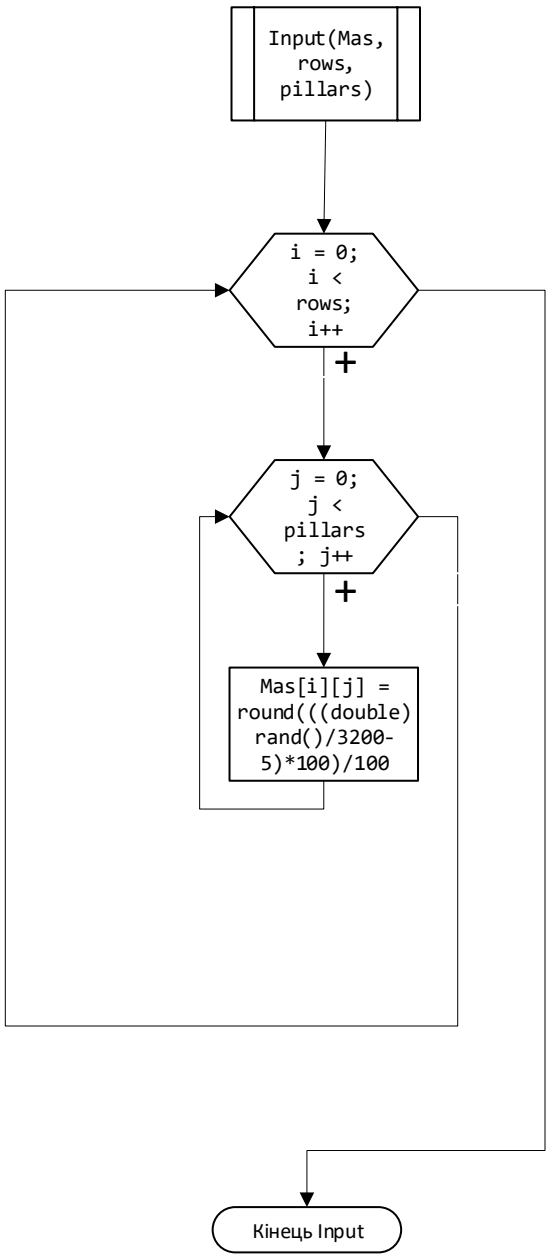
Блок-схеми алгоритму

Основна програма **main()**

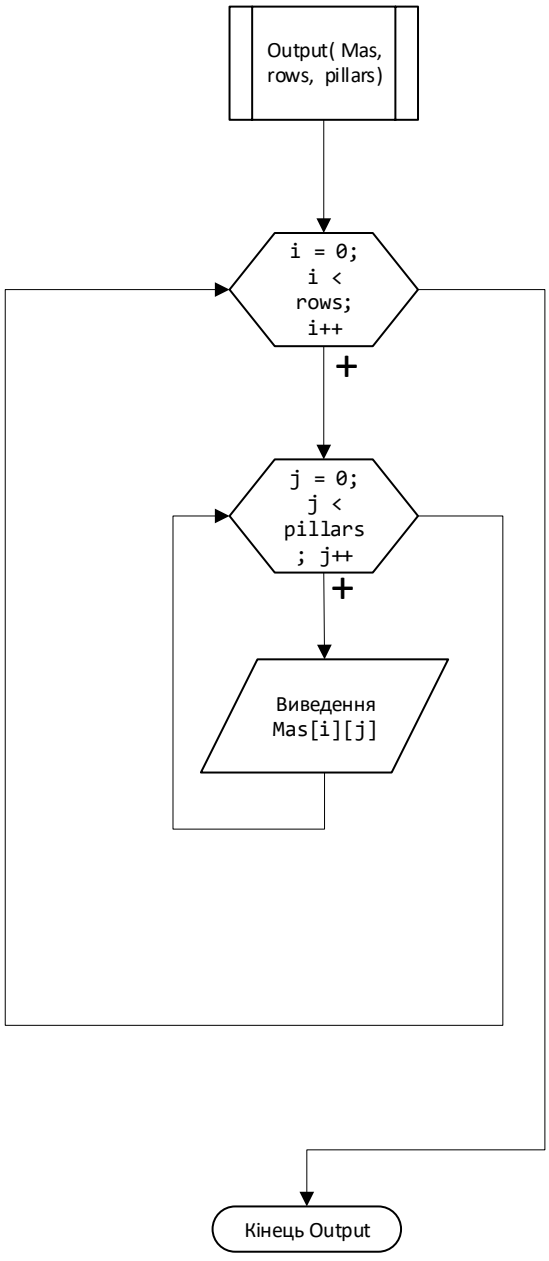


Підпрограми

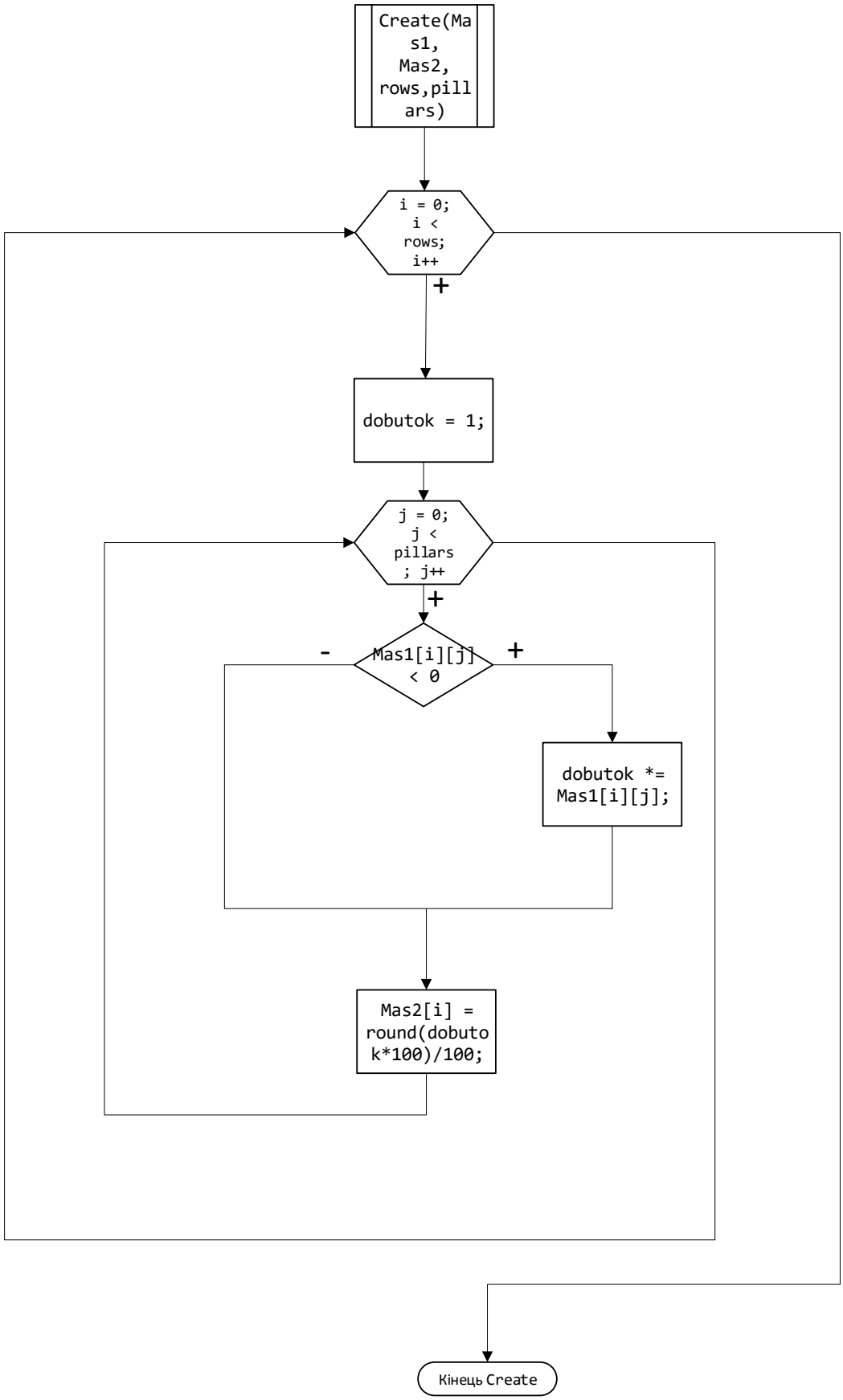
Input()



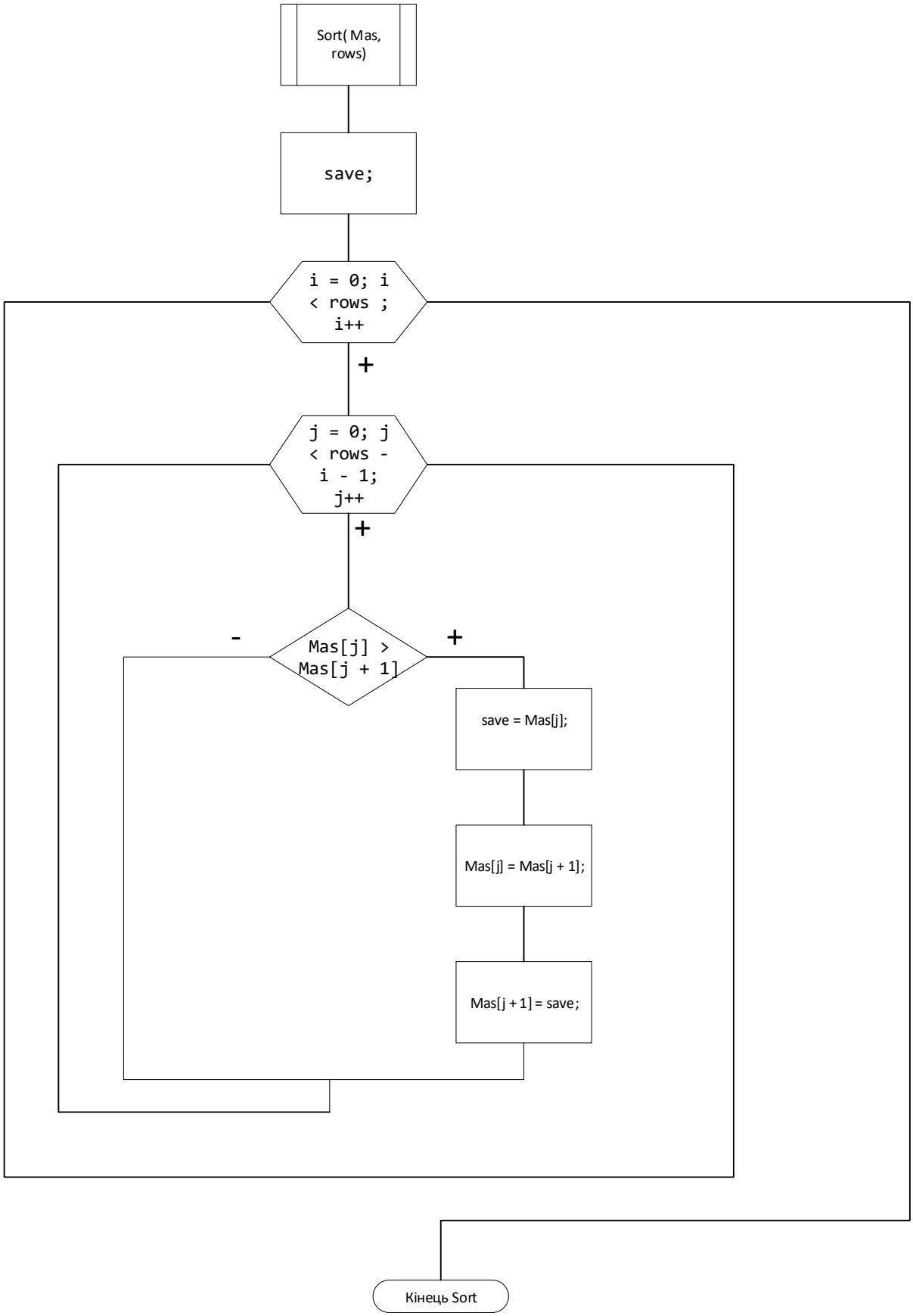
Output()


Output(Mas, rows, pillars)
i = 0; i < rows; i++
+
j = 0; j < pillars ; j++
+
Виведення Mas[i][j]
Кінець Output

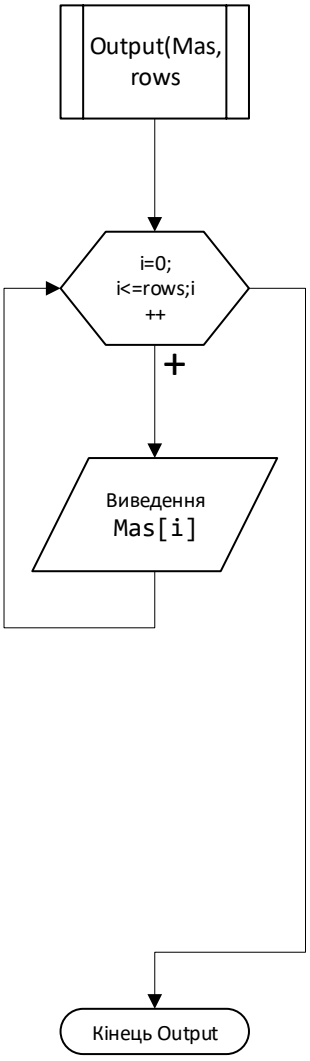
Create()


Create(Mas1, Mas2, rows,pillars)
i = 0; i < rows; i++
+
dobutok = 1;
j = 0; j < pillars ; j++
+
Mas1[i][j] < 0
-
+
dobutok *= Mas1[i][j];
Mas2[i] = round(dobutok*100)/100;
Кінець Create

Sort()


Sort(Mas, rows)
save;
i = 0; i < rows ; i++
+
j = 0; j < rows - i - 1; j++
+
Mas[j] > Mas[j + 1]
-
+
save = Mas[j];
Mas[j] = Mas[j + 1];
Mas[j + 1] = save;
Кінець Sort

Output()

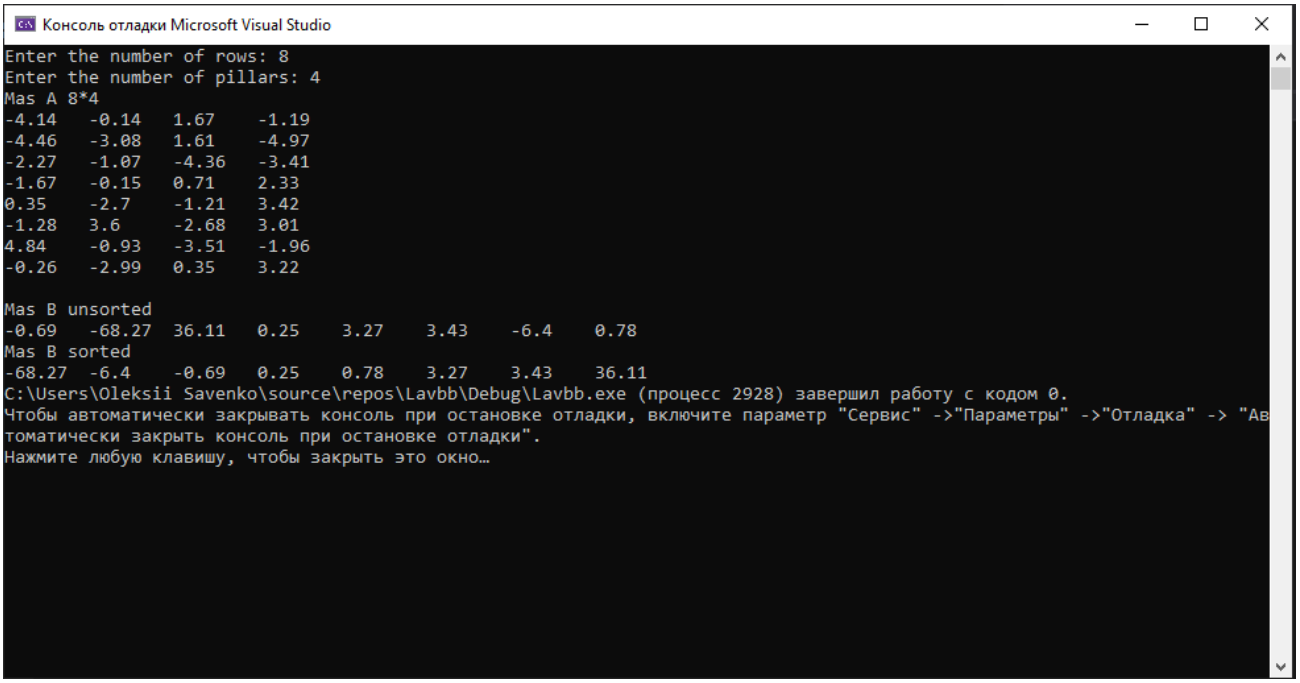


Код програми на мові C++:

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


void Generation(double**, int,int);
void Input(double**, int, int);
void Output(double**, int, int);
void Output(double*, int);
void Create(double**, double*, int,int);
void Sort(double*, int);
void Destruction(double**, int);

int main() {
	srand(time(NULL));
	int rows;
	cout << "Enter the number of rows: "; cin >> rows;   //Кількість рядків масиву

	int pillars;
	cout << "Enter the number of pillars: "; cin >> pillars;


	double** A = new double* [rows];

	double* B = new double[rows];

	Generation(A, rows, pillars);
	Input(A, rows, pillars);
	cout << "Mas A 8*4" << endl;
	Output(A, rows, pillars);

	Create(A, B, rows,pillars);
	cout << endl<<"Mas B unsorted" << endl;
	Output(B, rows);
	Sort(B, rows);
	cout << endl<<"Mas B sorted"<<endl;
	Output(B, rows);
	Destruction(A, rows);
	delete [] B;
}



void Generation(double** Mas, int rows, int pillars) {
	for (int i = 0; i < rows; i++) {
		Mas[i] = new double[pillars];

	}
}
void Input(double**Mas, int rows, int pillars) {
	for (int i = 0; i < rows; i++) {
		for (int j = 0; j < pillars; j++) {
			Mas[i][j] = round(((double)rand()/3200-5)*100)/100;
		}
	}
}
void Output(double** Mas, int rows, int pillars) {
	for (int i = 0; i < rows; i++) {
```

```
			for (int j = 0; j < pillars; j++) {
				cout << Mas[i][j] << "\t";
			}
			cout << endl;
		}
}
void Create(double**Mas1, double* Mas2, int rows,int pillars) {
		for (int i = 0; i < rows; i++) {
			double dobutok = 1;
			for (int j = 0; j < pillars; j++) {
				if (Mas1[i][j] < 0) {
					dobutok *= Mas1[i][j];
				}

				Mas2[i] = round(dobutok*100)/100;
			}
		}
}
void Output(double* Mas, int rows) {
		for (int i = 0; i < rows; i++) {
			cout << Mas[i] << "\t";
		}
}
void Sort(double* Mas, int rows) {
		double save;
		for (int i = 0; i < rows ; i++) {
			for (int j = 0; j < rows - i - 1; j++) {
				if (Mas[j] > Mas[j + 1]) {
					save = Mas[j];
					Mas[j] = Mas[j + 1];
					Mas[j + 1] = save;
				}
			}

		}
}
void Destruction(double** A, int m) {
		for (int i = 0; i < m; i++) {
			delete[] A[i];
		}
}
```

Випробування алгоритму на мові С++:



Випробування алгоритму

Блок	Дія
	Початок
1. Заповнення матриці **Generation()**	**-3.48 2.17 -1.14 0.11** **2.29 2.38 -3.97 -2.91** **-2.07 5.15 -3.54 -4.74** **-0.34 -3.17 2.57 -4.45** **-4.59 -4.71 0.29 -1.78** **2.79 0.21 2.1 -2.09** **-1.98 2.7 4.01 1.22** **2.91 2.13 2.95 -3.52**

2. Створення одновимірного масиву на основі добутку від'ємних елементів відповідних рядків матриці **Create()**	**3.97 11.55 -34.73 -4.8 -38.48 -2.09 -1.98 -3.52**
3. Відсортування обміном елементів одновимірного масиву **Sort()**	**-38.48 -34.73 -4.8 -3.52 -2.09 -1.98 3.97 11.55**
	Кінець

Висновок

Я дослідив алгоритми пошуку та сортування, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Була розроблена постановка задачі, де була пояснена логіка алгоритму, математична модель основної програми та підпрограм, псевдокод де розписаний алгоритм, а також блок-схеми основної програми та підпрограм які застосовуються при роботі алгоритму. Для розв'язання завдання було створено матрицю **A()** розмірності 8 x 4, а після цього **Create()** одновимірний масив **B()** розмірності 8, елементами якого стали добутки від'ємних значень відповідних рядків двовимірного масиву. Після цього за допомогою сортування обміном було відсортовано одновимірний масив **B()** за зростанням, у результаті чого було отримано відсортований за зростанням одновимірний масив **B()** . За допомогою випробування алгоритму, я перевірив його вірність і довів, що він є вірним.

Отже, мій алгоритм та програмний код написаний на його основі можна використовувати для вирішення завдань даного типу.