

Основи програмування – 1. Алгоритми та структури даних

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 10

Виконав студент

ІІ-15, Закірова Олександра Володимирівна
(шифр, прізвище, ім'я, по батькові)

Перевірив

(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 8

Дослідження алгоритмів пошуку та сортування

Варіант 10

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

Завдання №10

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

1. Опису змінної індексованого типу (двовимірний масив) розмірністю 4 x 8.
2. Ініціювання змінної, випадковими дійсними значеннями.
3. Створення нової змінної індексованого типу (одновимірний масив) та її ініціювання мінімальними значеннями кожного з стовпців двовимірного масиву.
4. Сортування одновимірного масиву методом Шела за спаданням.

Постановка задачі. Результатом виконання завдання є відсортований за спаданням одновимірний масив. Для розв’язання потрібна лише розмірність двовимірного масиву. Інших початкових даних для розв’язку не потрібно.

Математична побудова. Складемо таблицю змінних.

Змінна	Тип	Ім'я	Призначення
Число i	Цілий	i	Проміжна змінна
Число j	Цілий	j	Проміжна змінна
Число $\text{Matrix}[4][8]$	Дійсний	Matrix	Проміжна змінна

minsArray[8]	Дійсний	minsArray	Проміжна змінна
min	Дійсний	min	Проміжна змінна
step	Дійсний	step	Проміжна змінна
tmp	Дійсний	tmp	Проміжна змінна
Результат minsArray[8]	масив	minsArray	Результат

Таким чином, математичне формулювання задачі зводиться до генерації випадкових елементів двовимірного масиву, перебору цього масиву у пошуку найменших елементів кожного стопчика та сортування множини цих елементів за спаданням(методом Шела).

Розв’язання:

Програмні специфікації запишемо у псевдокодi та графічній формi у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію перебору стовпців множини у пошуку мінімальних значень.

Крок 3. Деталізуємо використовані у програмі функції генерації випадкового двовимірного масиву заданої розмірності та сортування методом Шела.

Псевдокод

крок 1

початок

Ініціалізація матриці рандомними цілими числами

Перебір стовпців множини у пошуку мінімальних значень

Сортування множини мінімальних значень методом Шела за спаданням

кінець

крок 2

початок

fillRandomNumbers(int Matrix[4][8], 4, 8)

int minsArray[m]

int min

для **int j = 0, j < m, j++**

min = 0

для **int i = 0, i < n, i++**

якщо Matrix[min][j] < Matrix[i][j]

min = i

інакше все повторити

все повторити

minsArray[j] = Matrix[min][j]

sortShell(int 8, int minsArray[])

кінець

крок 2

початок

для **int i = 0, i < 4, i++**

int j = 0, j < 8, j++

Matrix[i][j] = rand()%10

все повторити

все повторити

int minsArray[8]

int min

для **int j = 0, j < 4, j++**

min = 0

для **int i = 0, i < 8, i++**

якщо $\text{Matrix}[\text{min}][j] < \text{Matrix}[i][j]$

$\text{min} = i$

інакше все повторити

все повторити

$\text{minsArray}[j] = \text{Matrix}[\text{min}][j]$

$\text{int step} = 8 / 2$

якщо $\text{step} > 0$

для $\text{int } i = 0, i < (8 - \text{step}), i++$

$\text{int } j = i$

якщо $(j \geq 0) \ \&\& \ (\text{arr}[j] < \text{arr}[j + \text{step}])$

$\text{int tmp} = \text{arr}[j]$

$\text{arr}[j] = \text{arr}[j + \text{step}]$

$\text{arr}[j + \text{step}] = \text{tmp}$

$j -= \text{step}$

все повторити

інакше $\text{step} /= 2$

все повторити

все повторити

Вивід minsArray

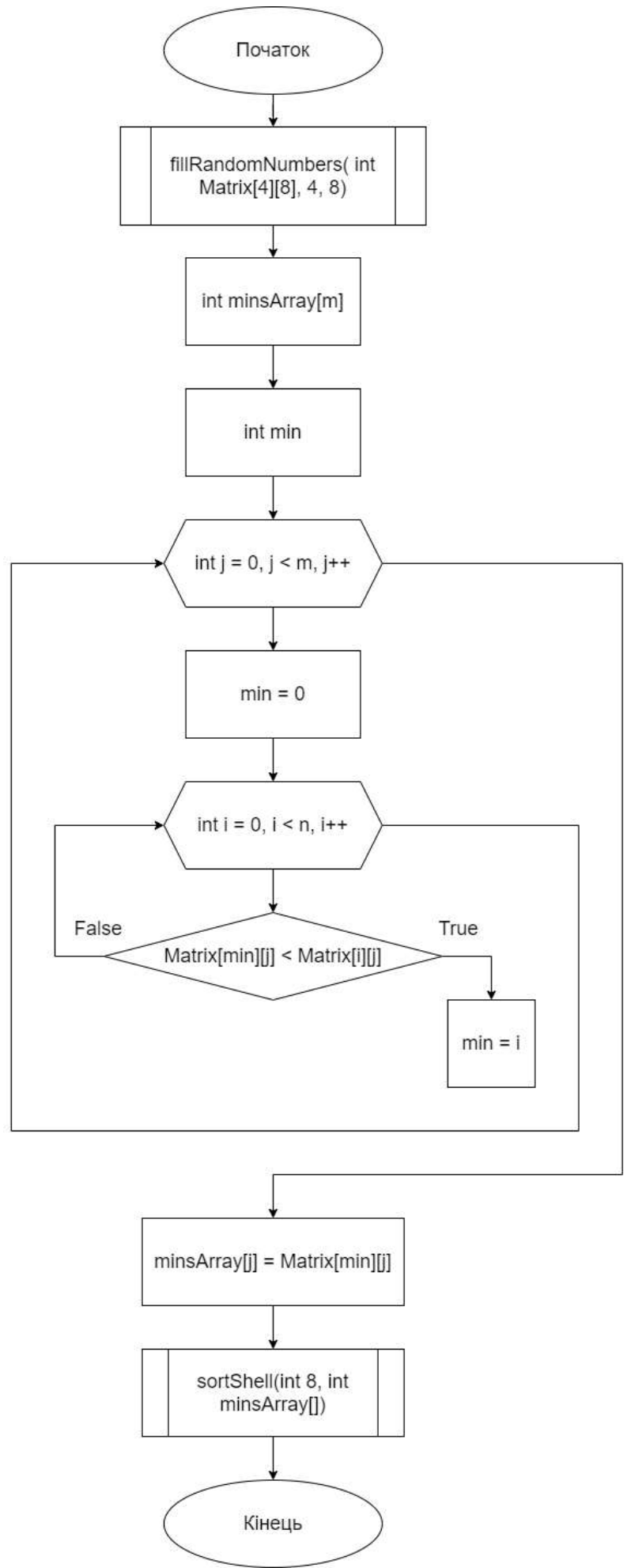
кінець

Блок-схема

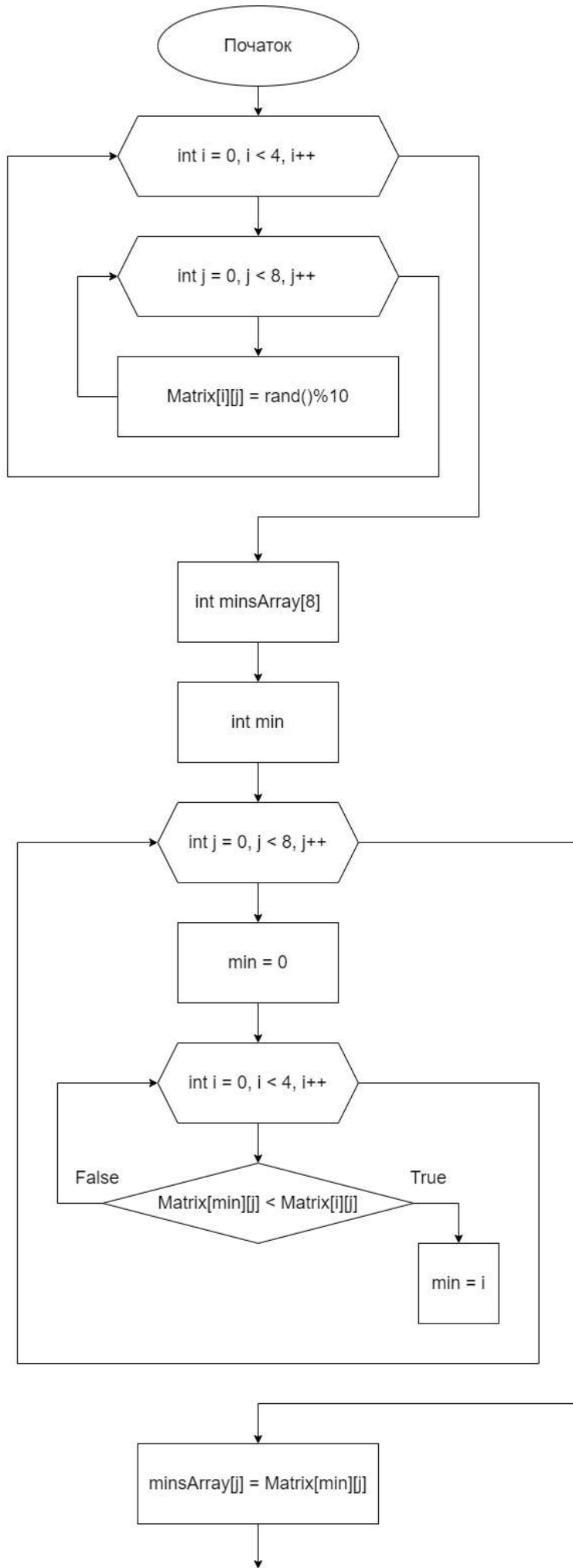
крок 1

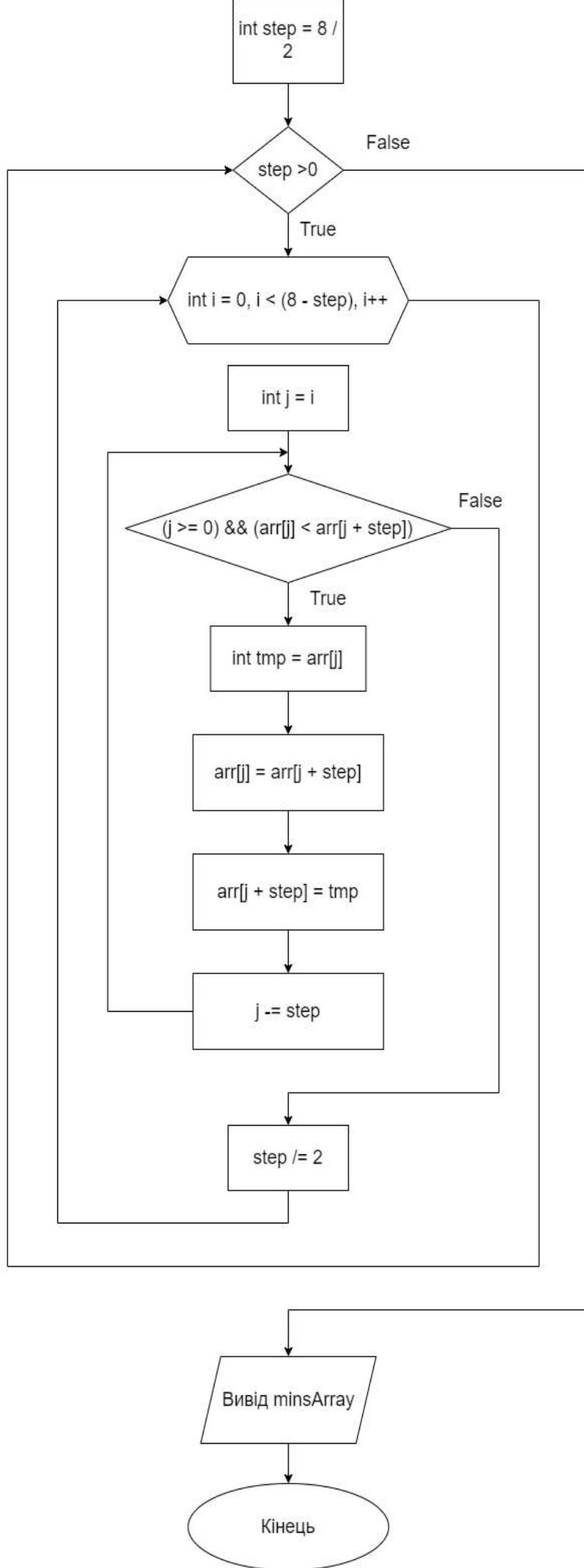


крок 2



крок 3





Код програми

ASDlab8 main.cpp

```
1  #include <iostream>
2  #include <ctime>
3
4  using namespace std;
5
6  void DeleteM(int **M, int numRows) {
7      for (size_t i = 0; i < numRows; ++i) {
8          delete[] M[i];
9      }
10     delete[] M;
11 }
12
13 void FillRandomNumbers(int **matrix, int numRows, int numCols) {
14     for (size_t row = 0; row < numRows; row++) {
15         for (size_t column = 0; column < numCols; column++) {
16             matrix[row][column] = rand() % 10; // NOLINT(cert-msc50-cpp)
17         }
18     }
19 }
20
21 void PrintMatrix(int **M, const string &name, int numRows, int numCols) {
22     cout << "\n" << name << endl;
23     for (int i = 0; i < numRows; ++i) {
24         for (int j = 0; j < numCols; ++j) {
25             cout << M[i][j] << ' ';
26         }
27         cout << endl;
28     }
29 }
30
31 void sortShell(int size, int arr[]) {...}
32
33 void PrintArray(const string &name, int *arr, int n) {...}
34
35 main
```

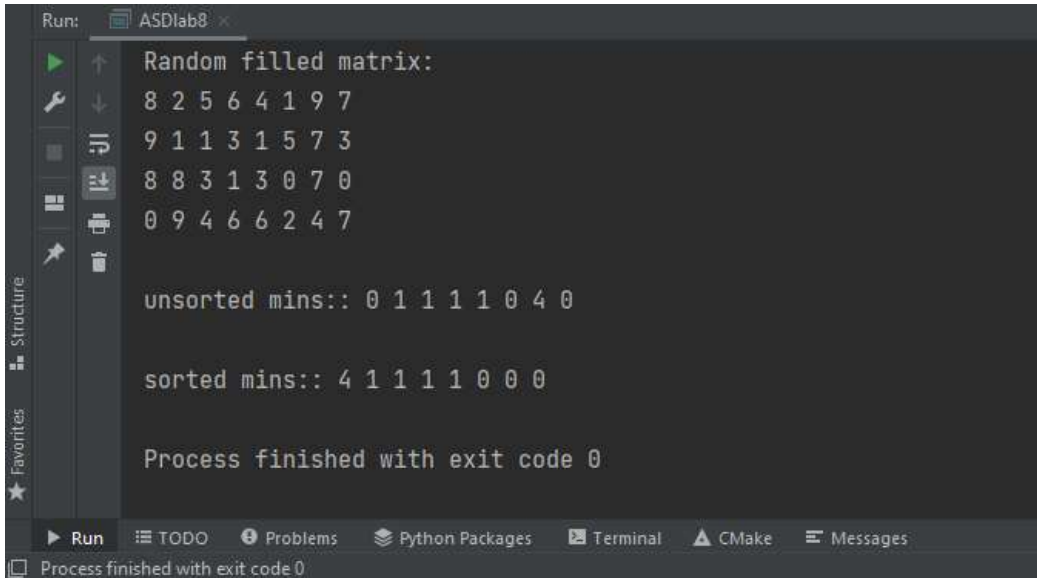
Run TODO Problems Python Packages Terminal CMake Messages

Build finished in 1 sec, 745 ms (a minute ago)

```
21 void PrintMatrix(int **M, const string &name, int numRows, int numCols) {
22     cout << "\n" << name << endl;
23     for (int i = 0; i < numRows; ++i) {
24         for (int j = 0; j < numCols; ++j) {
25             cout << M[i][j] << ' ';
26         }
27         cout << endl;
28     }
29 }
30
31 void sortShell(int size, int arr[]) {
32     int step = size / 2;
33     while (step > 0) {
34         for (int i = 0; i < (size - step); ++i) {
35             int j = i;
36             while ((j >= 0) && (arr[j] < arr[j + step])) {
37                 int tmp = arr[j];
38                 arr[j] = arr[j + step];
39                 arr[j + step] = tmp;
40                 j -= step;
41             }
42         }
43         step /= 2;
44     }
45 }
46
47 void PrintArray(const string &name, int *arr, int n) {
48     cout << "\n" << name << ": ";
49     for (int i = 0; i < n; ++i) {
50         cout << arr[i] << " ";
51     }
52     cout << endl;
53 }
54
```

```
54
55
56 ► int main() {
57     srand(time(nullptr));
58     int numRows = 4;
59     int numCols = 8;
60     //Инициализация случайными числами(от 0 до 10)
61     int **matrix = new int *[numRows];
62     for (size_t i = 0; i < numRows; ++i) {
63         matrix[i] = new int[numCols];
64     }
65     FillRandomNumbers(matrix, 4, 8);
66     PrintMatrix(matrix, "Random filled matrix:", 4, 8);
67     //Поиск минимальных элементов
68     int minsArr[8];
69     int min;
70     for (int j = 0; j < 8; j++) {
71         min = 0;
72         for (int i = 0; i < 4; i++) {
73             if (matrix[min][j] > matrix[i][j]) {
74                 min = i;
75             }
76         }
77         minsArr[j] = matrix[min][j];
78     }
79     PrintArray("unsorted mins:", minsArr, 8);
80     sortShell(8, minsArr);
81     PrintArray("sorted mins:", minsArr, 8);
82     DeleteM(matrix, 4);
83 }
84
```

Випробування алгоритму



Run: ASDlab8 x

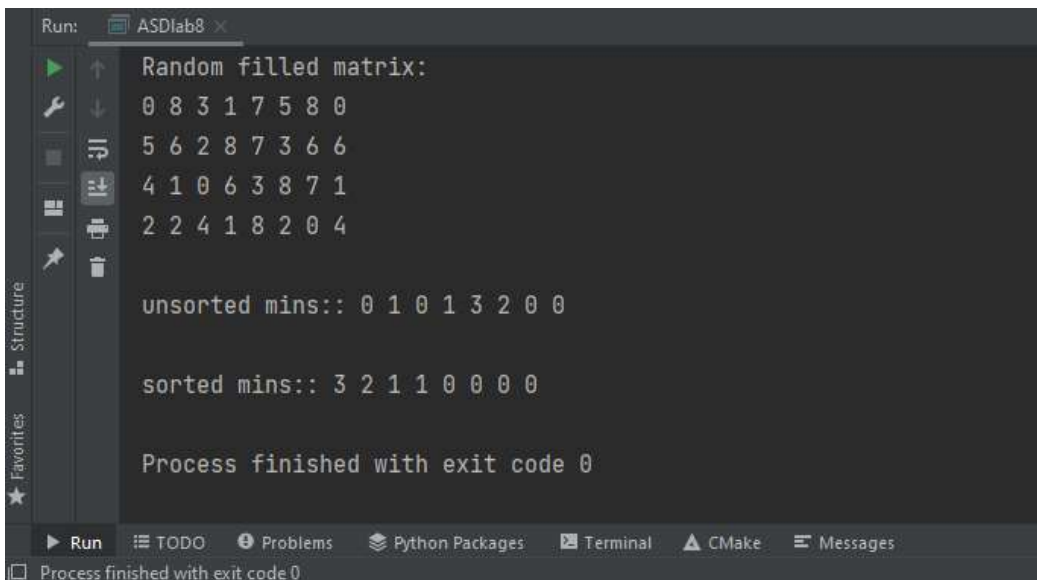
```
Random filled matrix:
8 2 5 6 4 1 9 7
9 1 1 3 1 5 7 3
8 8 3 1 3 0 7 0
0 9 4 6 6 2 4 7

unsorted mins:: 0 1 1 1 1 0 4 0

sorted mins:: 4 1 1 1 1 0 0 0

Process finished with exit code 0
```

Process finished with exit code 0



Run: ASDlab8 x

```
Random filled matrix:
0 8 3 1 7 5 8 0
5 6 2 8 7 3 6 6
4 1 0 6 3 8 7 1
2 2 4 1 8 2 0 4

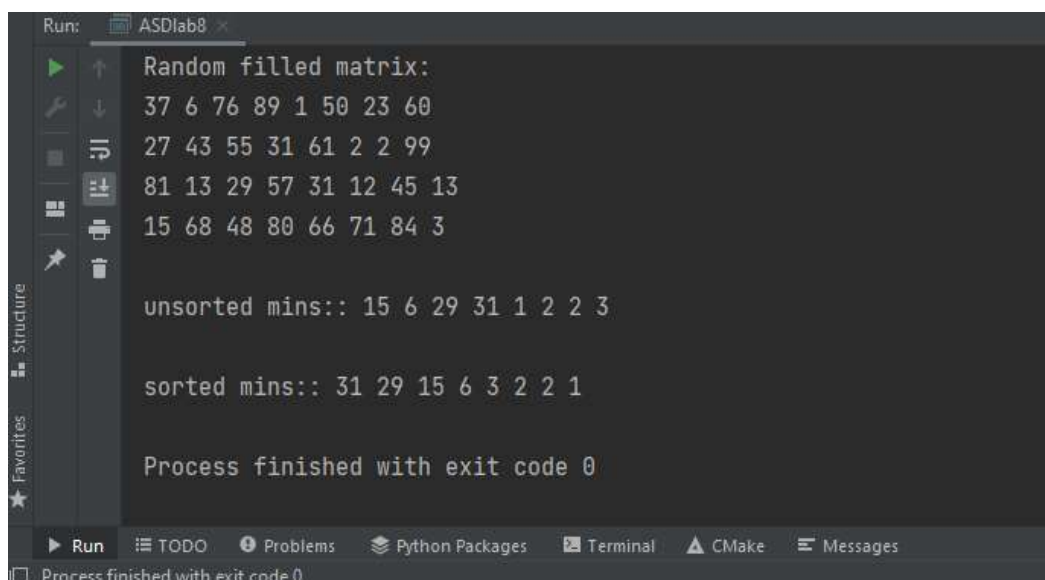
unsorted mins:: 0 1 0 1 3 2 0 0

sorted mins:: 3 2 1 1 0 0 0 0

Process finished with exit code 0
```

Process finished with exit code 0

(додаткова перевірка с генерацією елементів матриці у діапазоні сотні)



Run: ASDlab8 x

```
Random filled matrix:
37 6 76 89 1 50 23 60
27 43 55 31 61 2 2 99
81 13 29 57 31 12 45 13
15 68 48 80 66 71 84 3

unsorted mins:: 15 6 29 31 1 2 2 3

sorted mins:: 31 29 15 6 3 2 2 1

Process finished with exit code 0
```

Process finished with exit code 0

Висновок

Під час виконання лабораторної були досліджені особливості роботи алгоритмів пошуку та сортування та набуто практичних навичок їх використання під час складання програмних специфікацій. Покращено навички написання псевдокоду, побудови та тестування алгоритмів.