

# А л г о р и т м и   т а   с т р у к т у р и   д а н и х .

---

## О с н о в и   а л г о р и т м і з а ц і ї

*Додаток 1*

Міністерство освіти і науки України  
Національний технічний університет України «Київський політехнічний  
інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації  
і управління

Звіт

з лабораторної роботи №7  
з дисципліни «Алгоритми та структури даних-1.  
Основи алгоритмізації»  
«Дослідження лінійного пошуку в послідовностях»  
Варіант 6

Виконав  
студент

ІП-15, Волинець Кирило Михайлович

(шифр, прізвище, ім'я, по батькові)

Перевірів

Вечерковська Анастасія Сергіївна

( прізвище, ім'я, по батькові)

# А л г о р и т м и   т а   с т р у к т у р и   д а н и х .

## О с н о в и   а л г о р и т м і з а ц і ї

### Лабораторна робота 7

#### Дослідження лінійного пошуку в послідовностях

**Мета** - дослідити методи послідовного пошуку у впорядкованих і неупорядкованих послідовностях та набути практичних навичок їх використання під час складання програмних специфікацій.

### Індивідуальне завдання

#### Варіант 6

6	$73 - i$	$64 + 2 * i$	Кількість елементів між максимальним та мінімальним елементами
---	----------	--------------	--

### Побудова математичної моделі.

Складемо таблицю імен .

Змінна	Тип	Ім'я	Призначення
Перший масив	Символьний[10]	array1	Вхідні
Другий масив	Символьний[10]	array2	Вхідні
Масив рівних елементів	Символьний[10]	array3	Проміжні
Ітератор1	Цілочисельний	i	Проміжні
Ітератор2	Цілочисельний	j	Проміжні
Розмір третього масиву	Цілочисельний	array3_size	Результат
Побудова першого і другого масивів	Пустий (функція)	build_arrays(char*, char*)	Обчислення
Знаходження рівних елементів та їх кількості	Цілочисельний (функція)	find_equal_elements(char*, char*, char*)	Обчислення

Побудуємо два масиви array1 та array2 за допомогою циклу та ітератору. Обчислення будуть здійснені в функції build\_arrays. Дані виводяться за допомогою покажчиків. Перший масив буде відсортований за спаданням, а другий за зростанням, тому що вони задані арифметичною прогресією. Далі у функції find\_equal\_elements встановимо перший ітератор i у кінець першого масиву, а другий ітератор j у початок другого. Через відсортованість ми влаштуємо «перегони», коли значення двох масивів будуть «наздоганяти» один одного. У точках з рівними значеннями ми передаємо це значення у array3 та збільшуємо array3\_size на 1. Перегони зупиняються коли один із ітераторів досяг іншого кінця масиву. Оскільки array1 спадний, ми встановили ітератор у кінець та будемо віднімати 1 замість додавання. Оскільки array3 буде теж відсортованим, то у

## О с н о в и   а л г о р и т м і з а ц і ї

початку буде найменший елемент, а у кінці – найбільший. Тому відстань між ними дорівнює довжині масиву -2, виключаючи крайні елементи.

### Розв'язання

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

### Псевдокод

#### Основна програма:

початок

```
    build_arrays(array1, array2);  
    array3_size = find_equal_elements(array1, array2, array3);  
    Вивести (array3_size - 2)
```

кінець

#### Підпрограми:

```
build_arrays(char* array1, char* array2)
```

```
    повторити для i від 0 до i < 10:
```

```
        array1[i] = 73 - i;
```

```
        array2[i] = 64 + 2 * i;
```

```
    все повторити
```

кінець build\_arrays

```
find_equal_elements(char* arr1, char* arr2, char* arr3)
```

```
    array3_size = 0
```

```
    i = 9
```

```
    j = 0
```

```
    поки i >= 0 та j <= 9 повторити:
```

```
        якщо arr1[i] < arr2[j] то i = i - 1
```

```
        інакше якщо arr1[i] > arr2[j] то j = j + 1
```

```
        інакше:
```

```
            arr3[array3_size] = arr1[i]
```

```
            array3_size = array3_size + 1
```

```
            i = i - 1
```

```
            j = j + 1
```

```
        все інакше
```

```
    все повторити
```

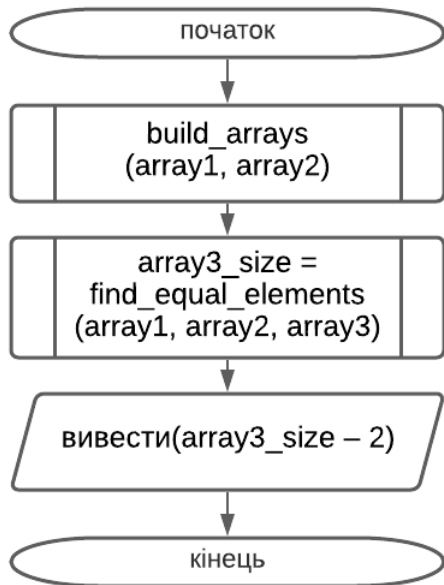
```
    повернути array3_size
```

кінець find\_equal\_elements

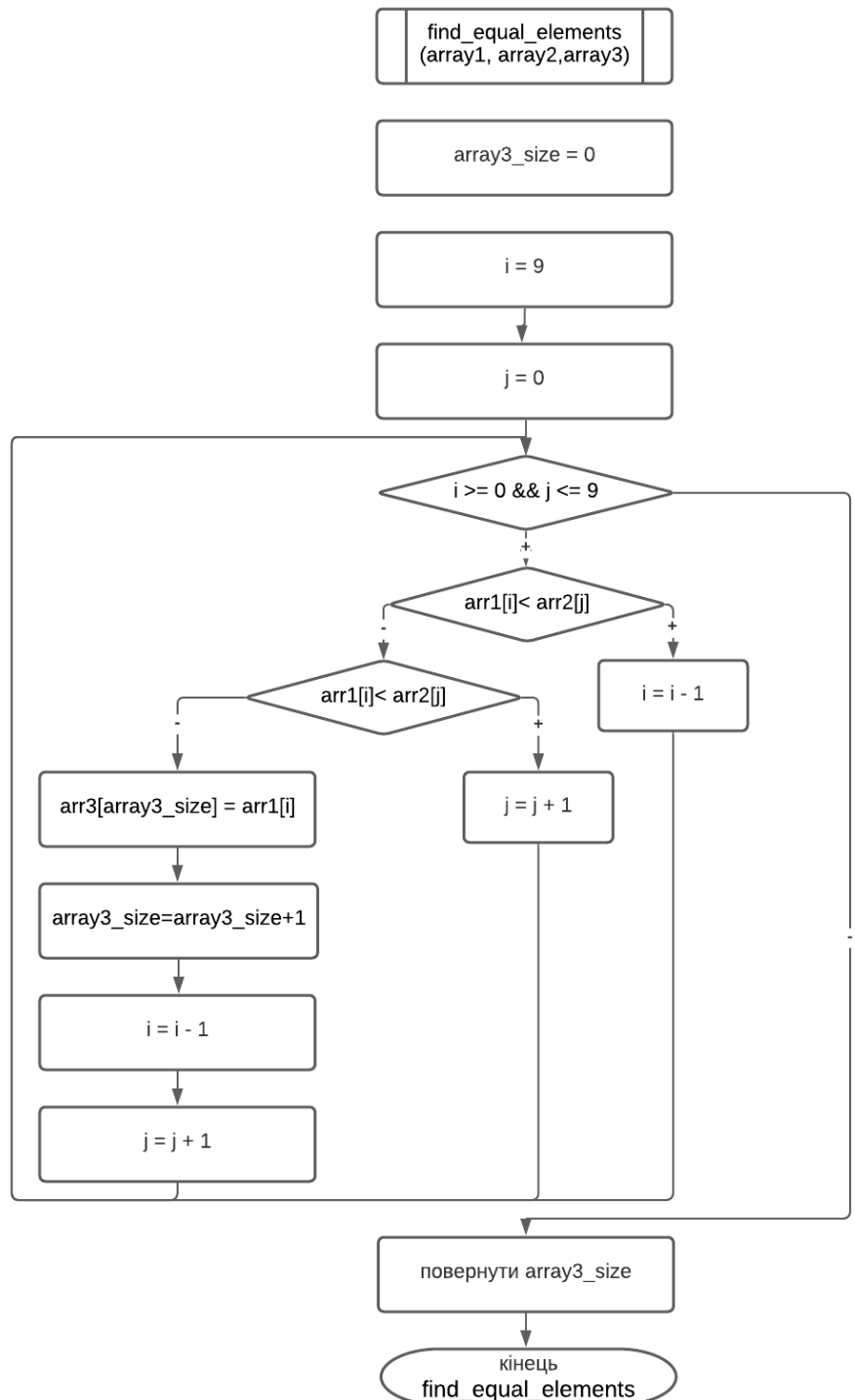
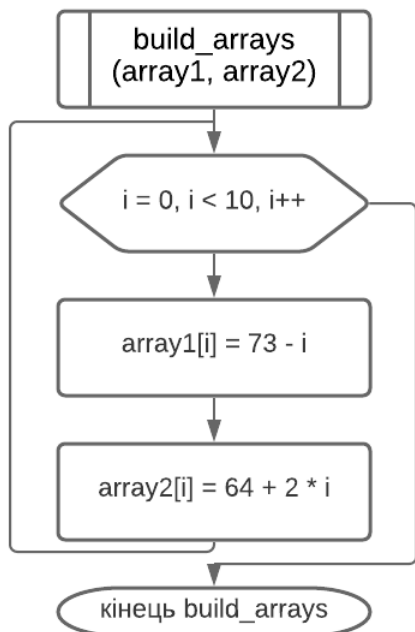
# Алгоритми та структури даних.

## Основи алгоритмізації

### Блок-схема:



### Підпрограми:



# Алгоритми та структури даних.

## Основи алгоритмізації

### Код програми

```
1  #include <iostream>
2  using namespace std;
3
4  void build_arrays(char*, char*);
5  int find_equal_elements(char*, char*, char*);
6
7  int main()
8  {
9      char array1[10], array2[10];
10     build_arrays(array1, array2);
11     char array3[10];
12     int array3_size = find_equal_elements(array1, array2, array3);
13     cout << array3_size - 2;
14 }
15
16 void build_arrays(char* arr1, char* arr2) {
17     for (int i = 0; i < 10; i++) {
18         arr1[i] = 73 - i;
19         arr2[i] = 64 + 2 * i;
20     }
21 }
22
23 int find_equal_elements(char* arr1, char* arr2, char* arr3) {
24     int array3_size = 0;
25     int i = 9, j = 0;
26     while (i >= 0 and j <= 10) {
27         if (arr1[i] < arr2[j]) i--;
28         else if (arr1[i] > arr2[j]) j++;
29         else {
30             arr3[array3_size] = arr1[i];
31             array3_size++;
32             i--;
33             j++;
34         }
35     }
36     return array3_size;
37 }
38
```

### Випробування коду

```
Перший масив:
I(73) H(72) G(71) F(70) E(69) D(68) C(67) B(66) A(65) @(64)
Другий масив:
@(64) B(66) D(68) F(70) H(72) J(74) L(76) N(78) P(80) R(82)
Третій масив:
@(64) B(66) D(68) F(70) H(72)
Кількість елементів між мінімальним та максимальним = 3
C:\Users\kiril\source\repos\ASD7\Debug\ASD7.exe (процесс 24808) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "А
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

### Висновки

Ми дослідили методи послідовного пошуку у впорядкованих і не впорядкованих послідовностях та набули практичних навичок їх використання під час складання програмних специфікацій. В результаті виконання лабораторної роботи ми отримали алгоритм для визначення двох масивів за формулою та знаходження кількості елементів між максимальним та мінімальним значень серед спільних членів обох масивів. В процесі випробування програма вивела три масиви та відповідь, що є вірною.