

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 № 7 з дисципліни «Алгоритми та  
структури даних-1. Основи алгоритмізації»

«Дослідження рекурсивних алгоритмів»

Варіант 11

Виконав студент ІП-13, Дем'янчук Олександр Петрович  
(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська Анастасія Сергіївна  
(прізвище, ім'я, по батькові)

## Лабораторна робота 7

### Дослідження рекурсивних алгоритмів

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

#### Індивідуальне завдання

#### Варіант 11

#### Завдання

№	Вираз для обчислення елемента		Знайти
	1-го масиву	2-го масиву	
11	$2 * i + 40$	$52 - 2 * i$	Елемент, який має максимальний код

#### 1. Постановка задачі

Методом рекурсивного алгоритму маємо обрахувати добуток 4 елементів геометричної прогресії, що зростає, для таких значень:  $b_1 = 5$ ,  $q = 3$ ,  $n = 4$ ;

## 2. Математична модель

Побудуємо таблицю імен змінних:

<i><b>Змінна</b></i>	<i><b>Тип</b></i>	<i><b>Ім'я</b></i>	<i><b>Призначення</b></i>
Перший масив	Символьний	array1	Проміжні дані
Другий масив	Символьний	array2	Проміжні дані
Третій масив	Символьний	array3	Проміжні дані
Лічильник	Цілий	i	Проміжні дані
Лічильник	Цілий	j	Проміжні дані
Лічильник	Цілий	k	Проміжні дані
Розмір масивів	Цілий	n	Проміжні дані
Максимальний код елемента	Цілий	max_code	Проміжні дані
Елемент з максимальним кодом	Цілий	maximum	Проміжні дані

Таблиця функцій:

<i><b>Змінна</b></i>	<i><b>Ім'я</b></i>
Заповнення масивів	fill_arrays
Заповнення третього масиву	fill_array3
Пошук максимального коду	find_max
Виведення елементів масиву	disp_array

Заповнення першого та другого масивів відбувається через підпрограму **fill\_arrays**, що заповнюватиме **array1** і **array2** за арифметичним циклом від 0 елемента до n-1

Підпрограму **disp\_array** за допомогою арифметичного циклу виводитиме значення елементів масивів від 0 до n-1.

За допомогою підпрограми **fill\_array3** реалізуємо заповнення третього масиву шляхом пошуку рівних елементів з першого та другого масивів, використовуючи два арифметичні цикли з лічильниками **i**, **j**, де **i** - лічильник першого масиву (від 0 до n-1), **j** - лічильник другого масиву (зі значенням від 0 до n-1). Проходячи по елементах масивів **array1** і **array2**, підпрограма буде за допомогою вкладеного циклу перевіряти рівність значень кодів елементів першого та другого масиву. Якщо рівність виконується, то підпрограма присвоїть значення цього елемента з першого масиву до третього масиву, визначивши індекс елемента за значенням лічильника **k** (від 0 до n-1) і

буде додавати то значення лічильника **k** одиницю після знаходження і внесення такого елемента до третього масиву

За допомогою підпрограми **find\_max** буде знайдено максимальне значення серед кодів елементів третього масиву. Підпрограма реалізована через простий цикл, який буде порівнювати значення лічильника **i** зі значенням **n** та за допомогою порівняння поточного та минулого значень елементів масиву знаходити відповідно мінімальне та максимальне значення.

### Розв'язання:

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

Крок 1. Визначимо основні дії;

Крок 2. Присвоюємо змінній **n** значення довжини масиву;

Крок 3. Деталізуємо підпрограму заповнення першого та другого масивів **fill\_arrays** та її виклик;

Крок 4. Деталізуємо підпрограму виведення масивів **disp\_array** та її виклик;

Крок 5. Деталізуємо підпрограму пошуку та заповнення елементів третього масиву (**fill\_array3**) та її виклик;

Крок 6. Деталізуємо підпрограму знаходження елемента третього масиву з максимальним значенням коду (**find\_max**) та її виклик;

Крок 7. Виводимо елемент з максимальним кодом

**Псевдокод:**

*Крок 1*

**початок**

присвоєння значення змінній **n**, оголошення масивів

виклик підпрограми **fill\_arrays** для заповнення першого та другого масивів

виклик підпрограми **disp\_array** двічі для виведення першого та другого масивів

виклик підпрограми **fill\_array3** для заповнення третього масиву

виклик підпрограми **find\_max** для знаходження елементу третього масиву з  
максимальним кодом елементу

виведення **max**

**кінець**

*Крок 7*

**початок**

```
n := 10  
array1[n]  
array2[n]  
array3[n]  
fill_arrays(array1, array2, n)  
disp_array(array1, n)  
disp_array(array2, n)  
fill_array3(array1, array2, array3, n)  
maximum := find_max(array3, n)  
виведення maximum
```

**кінець**

**підпрограма** **fill\_arrays**(**arr1**, **arr2**, **size**)

**повторити для i від 0 до size з кроком 1**

**arr1**[**i**] := 2\*i + 40

**arr2**[**i**] := 52 - 2\*i

**все повторити**

**все підпрограма**

**підпрограма** **disp\_array**(**arr**, **size**)

**повторити для i від 0 до size з кроком 1**

виведення **arr**[**i**]

**все повторити**

**все підпрограма**

**підпрограма** fill\_array3(arr1, arr2, arr3, size)

    k := 0

**повторити** для i від 0 до size з кроком 1

**повторити** для j від 0 до size з кроком 1

**якщо** arr1[i] == arr2[j]

**то**

                    arr3[k] := arr1[i]

                    k := k + 1

**все якщо**

**все повторити**

**все повторити**

**все підпрограма**

**підпрограма** find\_max(arr, size)

    i := 1

    max := arr[0]

**поки** ((arr[i] != 0) && (i < size)) **повторити**

**якщо** arr[i] > max

**то**

                max := array[i]

**все якщо**

        i := i + 1

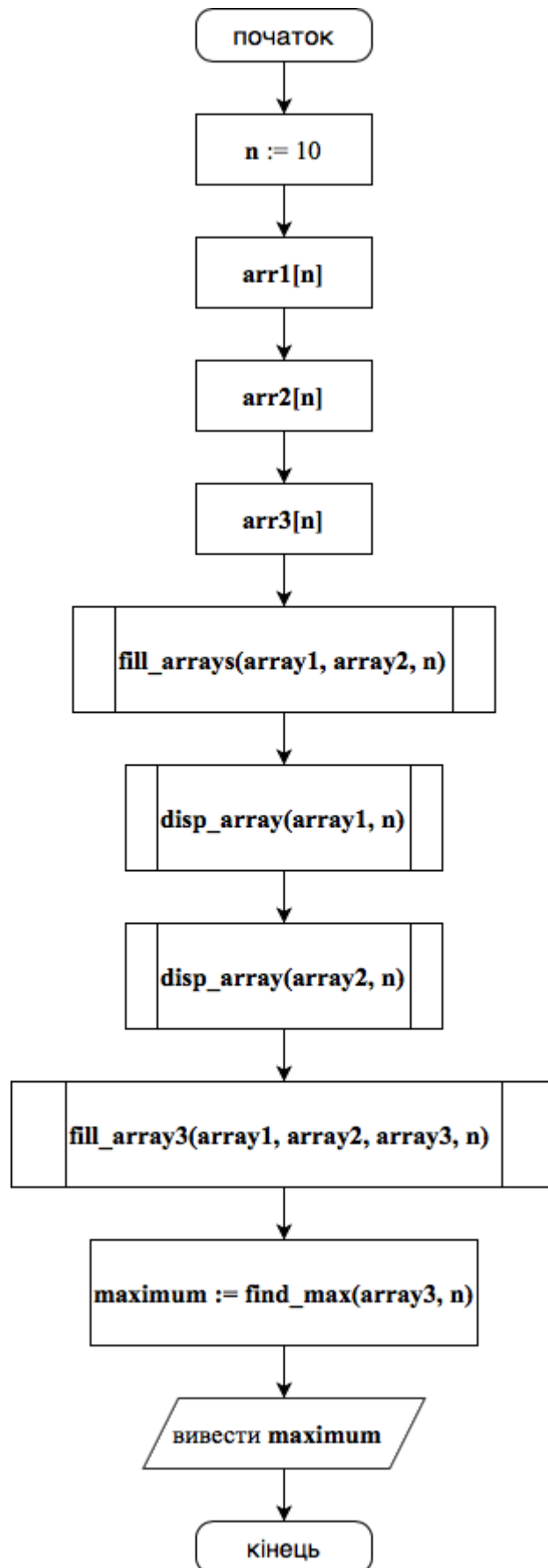
**все повторити**

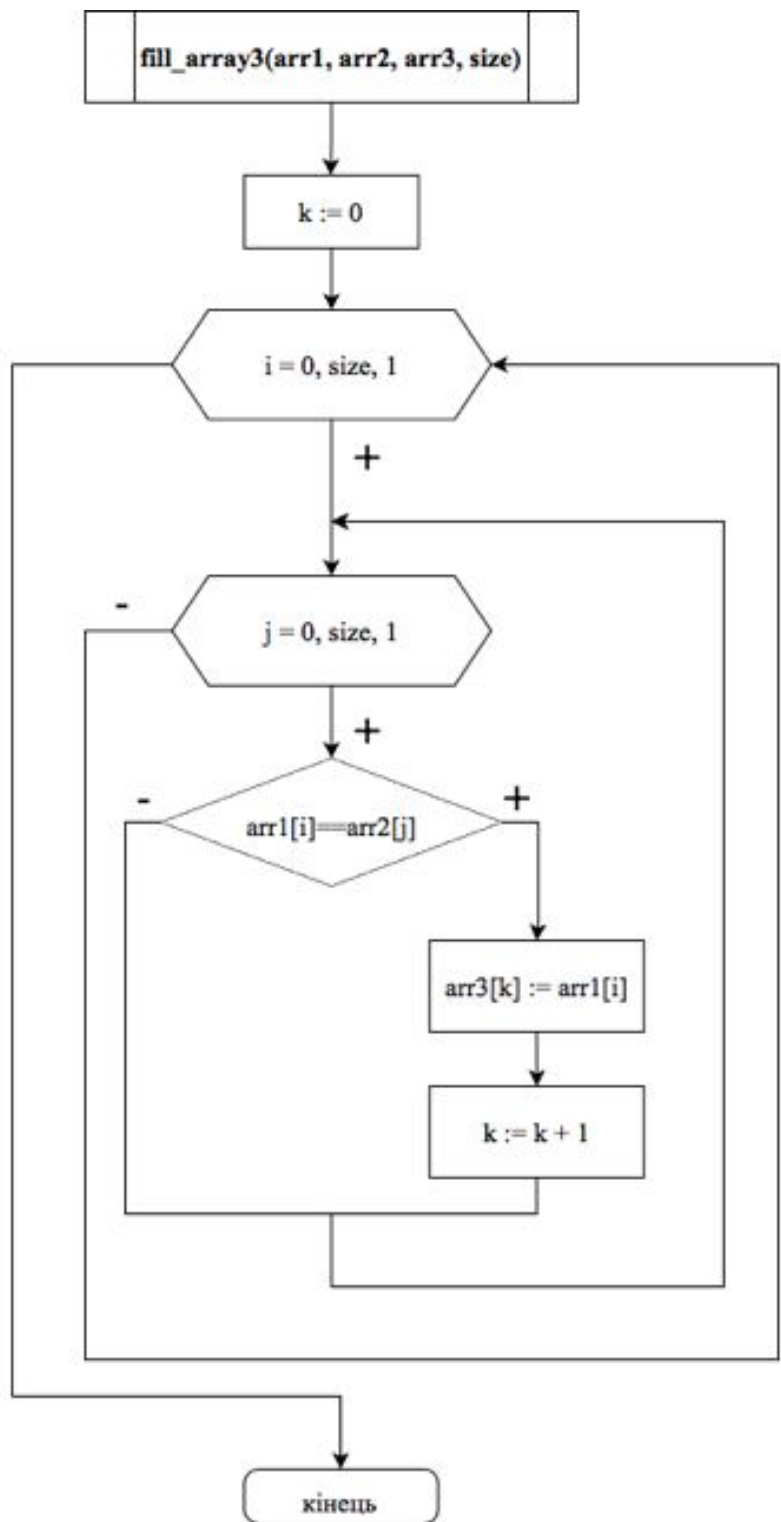
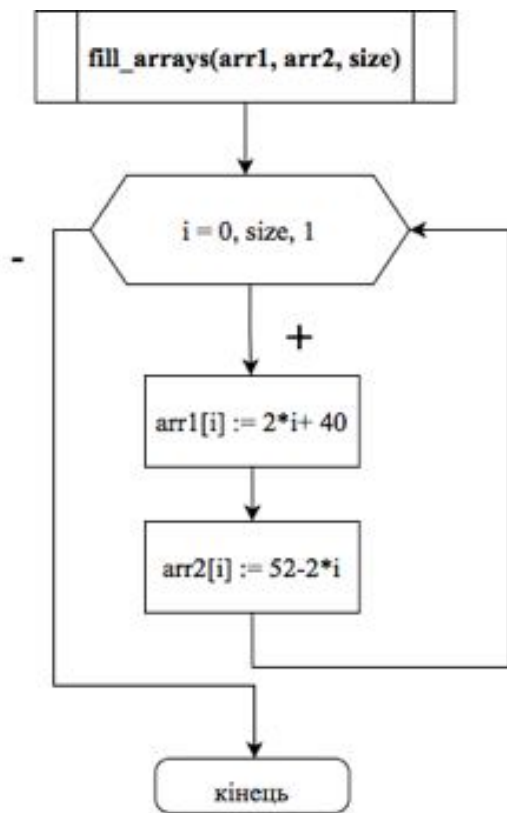
**return** max

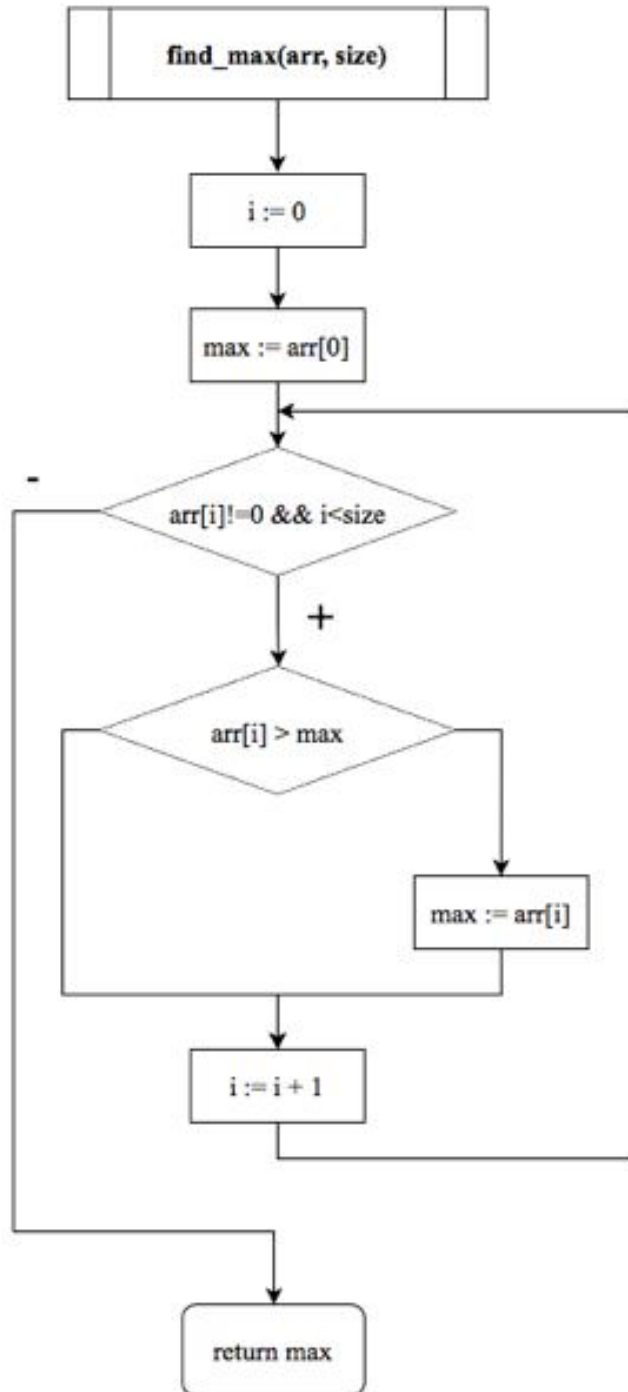
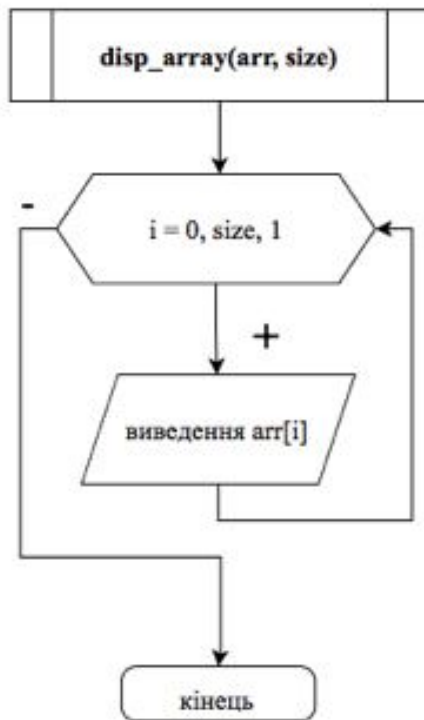
**все підпрограма**



## Блок-схема







## Програмний код

```
void fill_arrays(char[], char[], int);
void disp_array(char[], int);
void fill_array3(char[], char[], char[], int);
char find_max(char[], int);

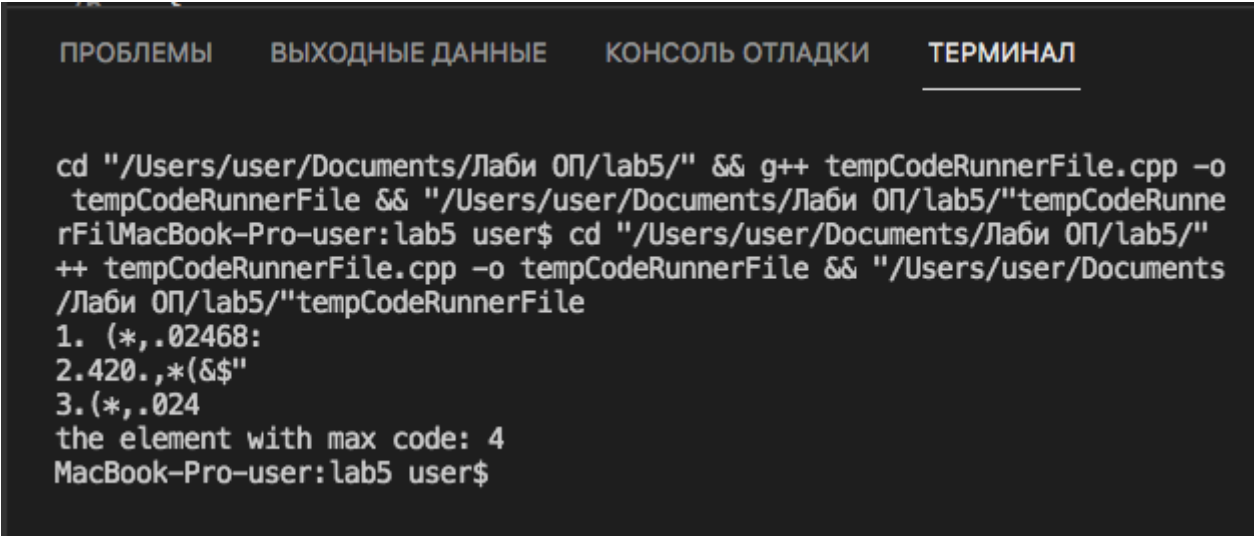
int main()
{
    const int n = 10;
    char array1[n];
    char array2[n];
    char array3[n];
    fill_arrays(array1, array2, n);
    cout << "1. ";
    disp_array(array1, n);
    cout << "2. ";
    disp_array(array2, n);
    cout << "3. ";
    fill_array3(array1, array2, array3, n);
    disp_array(array3, n);
    char maximum = find_max(array3, n);
    cout << "the element with max code: " << maximum << endl;
}
```

```
void fill_arrays(char arr1[], char arr2[], int size)
{
    for(int i = 0; i < size; i++)
    {
        arr1[i] = 2*i+40;
        arr2[i] = 52-2*i;
    }
}

void disp_array(char arr[], int size)
{
    for(int i = 0; i < size; i++)
    {
        cout << arr[i];
    }
    cout << endl;
}

void fill_array3(char arr1[], char arr2[], char arr3[], int size)
{
    int k = 0;
    for(int i = 0; i < size; i++)
    {
        for(int j = 0; j < size; j++)
        {
            if(arr1[i] == arr2[j])
            {
                arr3[k] = arr1[i];
                k++;
            }
        }
    }
}

char find_max (char arr[], int size)
{
    int i = 1;
    int charcode = (int)arr[0];
    char max = arr[i];
    while ((int)arr[i] != 0 && i < size)
    {
        if((int)arr[i] > charcode)
        {
            charcode = (int)arr[i];
            max = arr[i];
        }
    }
}
```



```
ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ

cd "/Users/user/Documents/Лаби ОП/lab5/" && g++ tempCodeRunnerFile.cpp -o
tempCodeRunnerFile && "/Users/user/Documents/Лаби ОП/lab5/"tempCodeRunne
rFilMacBook-Pro-user:lab5 user$ cd "/Users/user/Documents/Лаби ОП/lab5/"
++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile && "/Users/user/Documents
/Лаби ОП/lab5/"tempCodeRunnerFile
1. (*,.02468:
2.420.,*(&$"
3.(*,.024
the element with max code: 4
MacBook-Pro-user:lab5 user$
```

## Висновок

На лабораторній роботі дослідив методи послідовного пошуку у впорядкованих і неупорядкованих послідовностях та набув практичних навичок їх використання під час складання програмних специфікацій.