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

«Дослідження лінійного пошуку в
послідовностях»

Варіант 13

Виконав студент ІП-13, Жмайло Дмитро Олександрович
(шифр, прізвище, ім'я, по батькові)

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

Київ 2021

Лабораторна робота 6

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 13

№	Вираз для обчислення елемента		Знайти
	1-го масиву	2-го масиву	
13	$62 + 3 * i$	$74 - i$	Суму кодів мінімального та максимального елементів

Постановка задачі

Необхідно ініціалізувати 3 масиви символьних типів, які містять 10 елементів кожен; використовуючи вирази для обчислення елементів першого та другого масивів та арифметичні цикли, заповнити 1 та 2 масив. Потім заповнити третій масив спільними елементами з 2 та 3 масивів і обчислити суму кодів мінімального та максимального елементів в ньому.

Побудова математичної моделі

Відповідно до умови складемо таблицю змінних:

<i>Змінна</i>	<i>Тип</i>	<i>Назва</i>	<i>Призначення</i>
Перший масив	Символьний	arr1	Проміжні дані
Другий масив	Символьний	arr2	Проміжні дані
Третій масив	Символьний	arr3	Проміжні дані
Розмірність масивів	Цілий	n	Початкові дані
Лічильник i	Цілий	i	Проміжні дані
Лічильник j	Цілий	j	Проміжні дані
Лічильник k	Цілий	k	Проміжні дані
Мінімальний код символу	Цілий	min_code	Проміжні дані
Максимальний код символу	Цілий	max_code	Проміжні дані
Змінна для пошуку мінімального та максимального елементів в підпрограмі	Цілий	prev_code	Проміжні дані (змінна в підпрограмі)
Сума мінімального коду та максимального коду символів третього масиву	Цілий	sum	Вихідні дані
Масив в підпрограмі	Символьний	arr	Проміжні дані (змінна в підпрограмі)
Масив 1 в підпрограмі	Символьний	arr_a	Проміжні дані (змінна в підпрограмі)
Масив 2 в підпрограмі	Символьний	arr_b	Проміжні дані (змінна в підпрограмі)
Розмірність масива в підпрограмі	Символьний	arr	Проміжні дані (змінна в підпрограмі)

Отже, ми будемо заповняти перший та другий масив за допомогою підпрограми **fill_arr**, яка буде за допомогою арифметичного циклу заповнювати масиви від **0-го** до **n-1-го** елементу.

Використаємо підпрограму **display_arr**, яка за допомогою арифметичного циклу буде виводити значення масивів по одному, виводячи елементи від **0** до **n-1**.

Далі за допомогою підпрограми **fill_third_arr** знайдемо елементи першого та другого масивів, які є рівними і занесемо їх у третій масив, використовуючи два арифметичні цикли з лічильниками **i**, **j**, де **i** - лічильник першого масиву (зі значенням від **0** до **n-1**), **j** - лічильник другого масиву (зі значенням від **0** до **n-1**). Проходячи по елементах масивів **1** і **2**, підпрограма буде за допомогою вкладеного циклу перевіряти рівність значення коду елемента першого масиву та другого масиву. Якщо рівність буде справджуватись, то підпрограма буде заносити значення цього елемента (беручи його з першого масиву) до третього масиву, беручи індекс елемента як значення 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 підпрограм **find_min_code** та **find_max_code** будемо знаходити максимальне і мінімальне значення кодів елементів третього масиву. Для обох підпрограм використаємо 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 псевдокодi та графічній формi у вигляді блок-схеми.

Крок 1. Визначимо основні дії;

Крок 2. Присвоюємо значення змінній довжини масиву;

Крок 3. Деталізуємо підпрограму знаходження елементів першого та другого масивів (**fill_arr**) та її виклик;

Крок 4. Деталізуємо підпрограму виведення масивів (**display_arr**) та її виклик;

Крок 5. Деталізуємо підпрограму знаходження та заповнення елементів третього масиву (**fill_third_arr**) та її виклик;

Крок 6. Деталізуємо підпрограму знаходження елементу третього масиву з максимальним значенням коду (**find_max_code**) та її виклик;

Крок 7. Деталізуємо підпрограму знаходження елементу третього масиву з мінімальним значенням коду (**find_min_code**) та її виклик;

Крок 8. Знаходження та виведення змінної **sum**.

Псевдокод:

Крок 1

початок

присвоєння значення змінній **n**, оголошення масивів

виклик підпрограми **fill_arr** для заповнення першого та другого масивів

виклик підпрограми **display_arr** двічі для виведення першого та другого масивів

виклик підпрограми **fill_third_arr** для заповнення третього масиву

виклик підпрограми **find_max_code** для знаходження елементу третього масиву з максимальним кодом елементу

виклик підпрограми **find_min_code** для знаходження елементу третього масиву з мінімальним значенням коду елементу

знаходження **sum**

Виведення **sum**

кінець

підпрограма fill_arr(arr1, arr2, size)

реалізація підпрограми

все підпрограма

підпрограма display_arr(arr, size)

реалізація підпрограми

все підпрограма

підпрограма fill_third_arr(arr_a, arr_b, arr3, size)

реалізація підпрограми

все підпрограма

підпрограма find_min_code(arr, size)

реалізація підпрограми

все підпрограма

підпрограма find_max_code(arr, size)

реалізація підпрограми

все підпрограма

Крок 2

початок

`n := 10`

`arr1[n]`

`arr2[n]`

`arr3[n]`

виклик підпрограми **fill_arr** для заповнення першого та другого масивів

виклик підпрограми **display_arr** двічі для виведення першого та другого масивів

виклик підпрограми **fill_third_arr** для заповнення третього масиву

виклик підпрограми **find_max_code** для знаходження елементу третього масиву з максимальним кодом елементу

виклик підпрограми **find_min_code** для знаходження елементу третього масиву з мінімальним значенням коду елементу

знаходження **sum**

Виведення **sum**

кінець

Крок 3

початок

`n := 10`

`arr1[n]`

`arr2[n]`

`arr3[n]`

`fill_arr(arr1, arr2, n)`

виклик підпрограми **display_arr** двічі для виведення першого та другого масивів

виклик підпрограми **fill_third_arr** для заповнення третього масиву

виклик підпрограми **find_max_code** для знаходження елементу
третього масиву з максимальним кодом елементу

виклик підпрограми **find_min_code** для знаходження елементу
третього масиву з мінімальним значенням коду елементу

знаходження **sum**

Виведення **sum**

кінець

Крок 4

початок

`n := 10`

`arr1[n]`

`arr2[n]`

`arr3[n]`

`fill_arr(arr1, arr2, n)`

`display_arr(arr1, n)`

`display_arr(arr2, n)`

виклик підпрограми **fill_third_arr** для заповнення третього масиву

виклик підпрограми **find_max_code** для знаходження елементу
третього масиву з максимальним кодом елементу

виклик підпрограми **find_min_code** для знаходження елементу
третього масиву з мінімальним значенням коду елементу

знаходження **sum**

Виведення **sum**

кінець

Крок 5

початок

n := 10

arr1[n]

arr2[n]

arr3[n]

fill_arr(arr1, arr2, n)

display_arr(arr1, n)

display_arr(arr2, n)

fill_third_arr(arr1, arr2, arr3, n)

виклик підпрограми **find_max_code** для знаходження елемента
третього масиву з максимальним кодом елемента

виклик підпрограми **find_min_code** для знаходження елемента
третього масиву з мінімальним значенням коду елемента

знаходження **sum**

Виведення **sum**

кінець

Крок 6

початок

n := 10

arr1[n]

arr2[n]

arr3[n]

fill_arr(arr1, arr2, n)

display_arr(arr1, n)

display_arr(arr2, n)

fill_third_arr(arr1, arr2, arr3, n)

max_code := find_max_code(arr3, n)

виклик підпрограми **find_min_code** для знаходження елемента
третього масиву з мінімальним значенням кода елемента

знаходження **sum**

Виведення **sum**

кінець

Крок 7

початок

```
n := 10  
arr1[n]  
arr2[n]  
arr3[n]  
fill_arr(arr1, arr2, n)  
display_arr(arr1, n)  
display_arr(arr2, n)  
fill_third_arr(arr1, arr2, arr3, n)  
max_code := find_max_code(arr3, n)  
min_code := find_min_code(arr3, n)
```

знаходження **sum**

Виведення **sum**

кінець

Крок 8

початок

`n := 10`

`arr1[n]`

`arr2[n]`

`arr3[n]`

`fill_arr(arr1, arr2, n)`

`display_arr(arr1, n)`

`display_arr(arr2, n)`

`fill_third_arr(arr1, arr2, arr3, n)`

`max_code := find_max_code(arr3, n)`

`min_code := find_min_code(arr3, n)`

`sum := max_code + min_code`

Виведення **sum**

кінець

підпрограма fill_arr(arr1, arr2, size)

повторити для i від 0 до size з кроком 1

arr1[i] := 62 + 3*i

arr2[i] := 74 - i

все повторити

все підпрограма

підпрограма display_arr(arr, size)

повторити для i від 0 до size з кроком 1

Виведення arr[i]

все повторити

все підпрограма

підпрограма fill_third_arr(arr_a, arr_b, 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 arr_a[i] == arr_b[j]

то

arr3[k] := arr_a[i]

k := k + 1

все якщо

все повторити

все повторити

все підпрограма

підпрограма find_min_code(arr, size)

i := 1

prev_code := arr[0]

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

якщо arr[i] < prev_code

то

 prev_code := arr[i]

все якщо

 i := i + 1

все повторити

return prev_code

все підпрограма

підпрограма find_max_code(arr, size)

i := 1

prev_code := arr[0]

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

якщо arr[i] > prev_code

то

 prev_code := arr[i]

все якщо

 i := i + 1

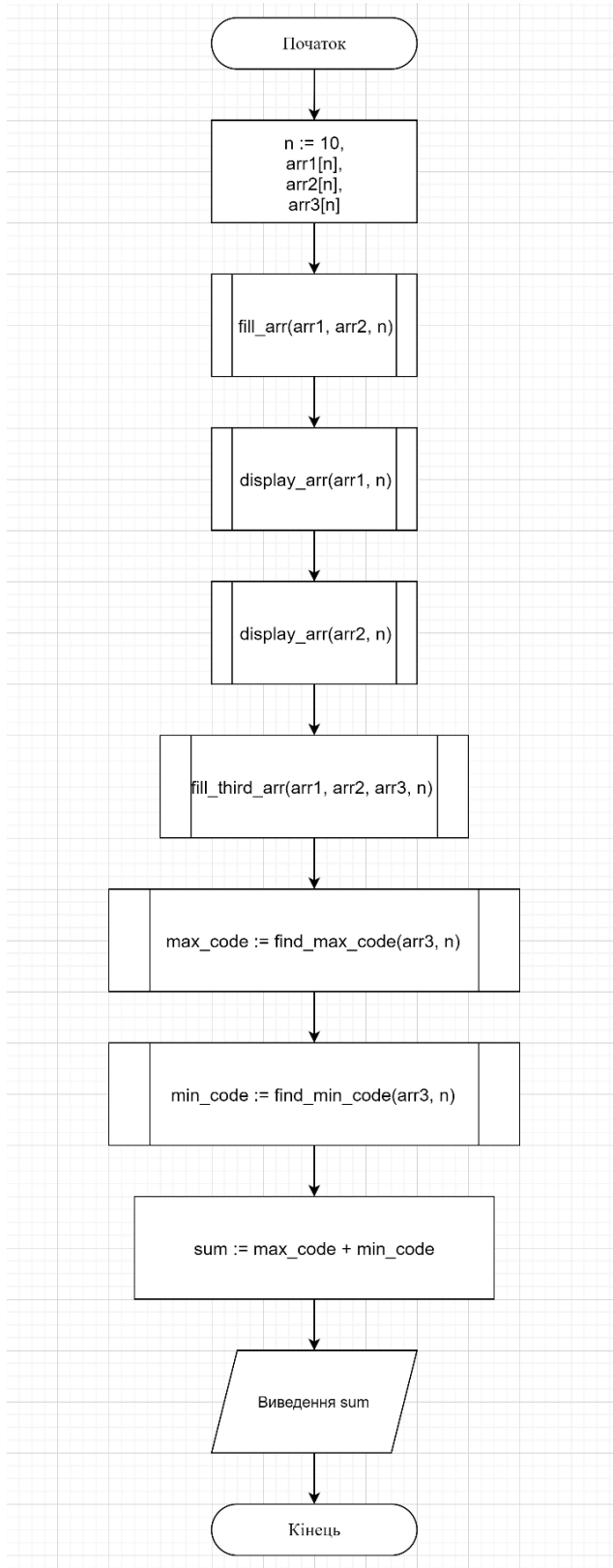
все повторити

return prev_code

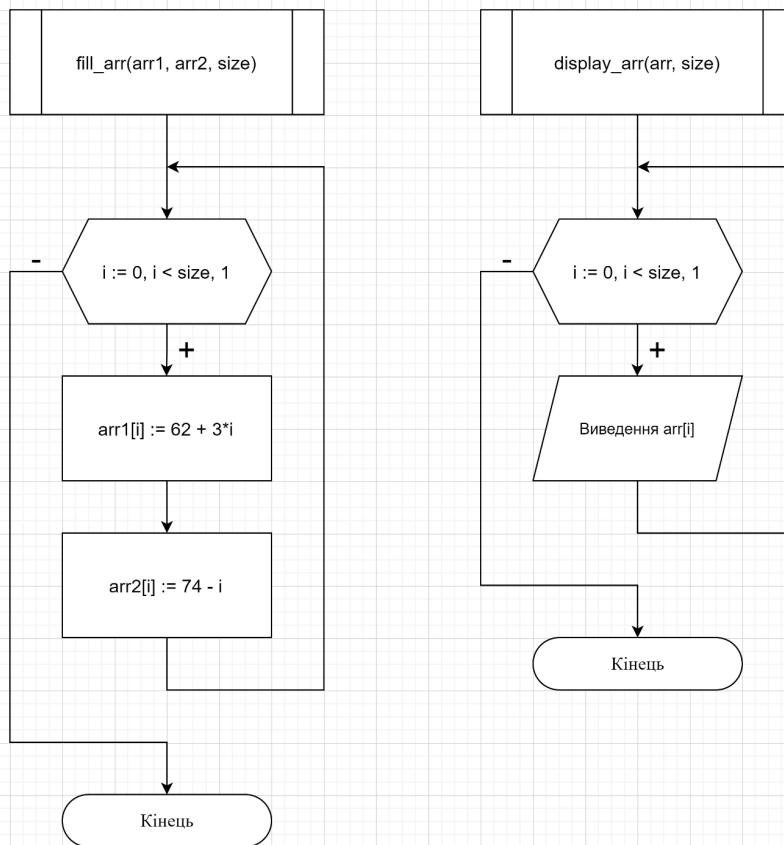
все підпрограма

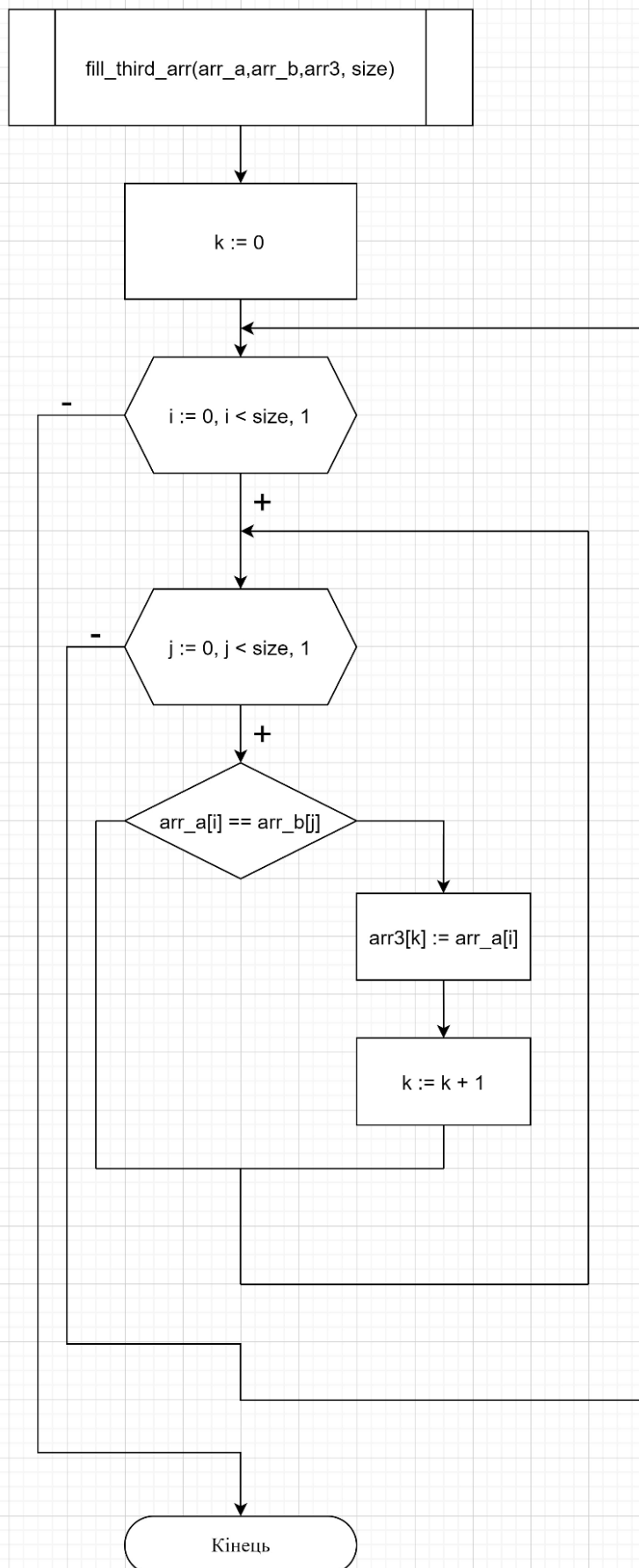
Блок-схема:

Основна програма:

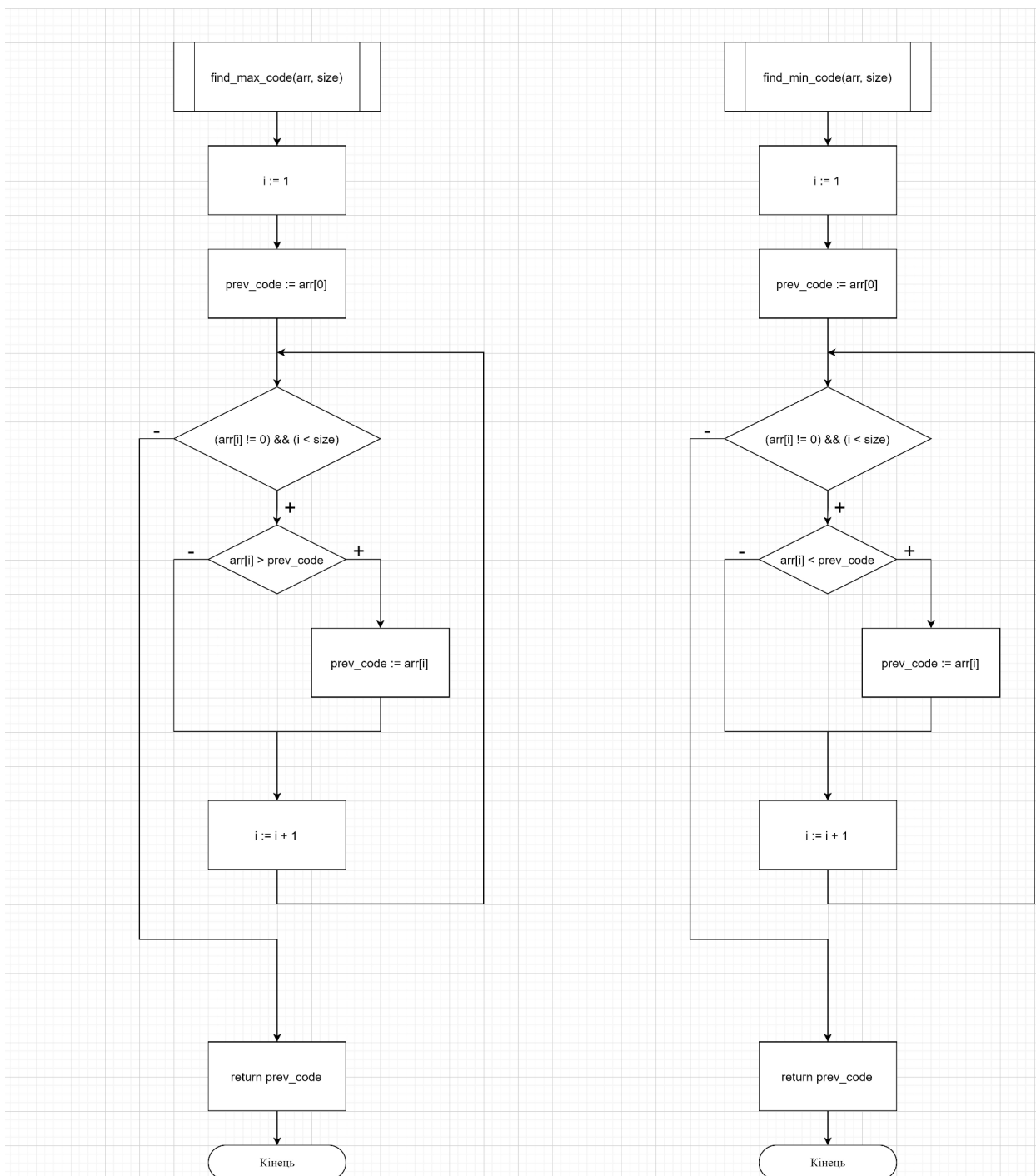


Підпрограми:





Основи програмування – 1. Алгоритми та структури даних



Код програми: (C++)

```
#include <iostream>
using namespace std;
void fill_arr(char[], char[], int);
void display_arr(char[], int);
void fill_third_arr(char[], char[], char[], int);
int find_max_code(char[], int);
int find_min_code(char[], int);
int main()
{
    const int n = 10;
    char arr1[n];
    char arr2[n];
    char arr3[n]{};
    int sum, min_code, max_code;
    fill_arr(arr1, arr2, n);
    cout << "Array 1:" << endl;
    display_arr(arr1, n);
    cout << "Array 2:" << endl;
    display_arr(arr2, n);
    fill_third_arr(arr1, arr2, arr3, n);
    cout << "Array 3:" << endl;
    display_arr(arr3, n);
    min_code = find_min_code(arr3, n);
    max_code = find_max_code(arr3, n);
    cout << "Min code is: " << min_code << '\n' << "Max code is: " << max_code << endl;
    sum = min_code + max_code;
    cout << "Sum: " << sum << endl;
    return 0;
}
```


```
void fill_arr(char arr1[], char arr2[], int size)
{
    for (int i = 0; i < size; i++)
    {
        arr1[i] = 62 + 3 * i;
        arr2[i] = 74 - i;
    }
}

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

```
void fill_third_arr(char arr_a[], char arr_b[], char arr3[], int size)
{
    int k = 0;
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            if (arr_a[i] == arr_b[j])
            {
                arr3[k] = arr_a[i];
                k++;
            }
        }
    }
}

int find_min_code(char arr[], int size)
{
    int i = 1;
    int prev_code = (int)arr[0];
    while ((int)arr[i] != 0 && i < size)
    {
        if ((int)arr[i] < prev_code)
        {
            prev_code = (int)arr[i];
        }
        i++;
    }
    return prev_code;
}
```

```
int find_max_code(char arr[], int size)
{
    int i = 1;
    int prev_code = (int)arr[0];
    while ((int)arr[i] != 0 && i < size)
    {
        if ((int)arr[i] > prev_code)
        {
            prev_code = (int)arr[i];
        }
        i++;
    }
    return prev_code;
}
```

 Консоль отладки Microsoft Visual Studio

```
Array 1:
> A D G J M P S V Y
Array 2:
J I H G F E D C B A
Array 3:
A D G J
Min code is: 65
Max code is: 74
Sum: 139
```


Висновок: На цій лабораторній роботі ми дослідили методи послідовного пошуку у впорядкованих і неупорядкованих послідовностях та набули практичних навичок їх використання під час складання програмних специфікацій. Навчилися роботи блок-схеми послідовностей та їх псевдокод. Склали програму на мові програмування C++; перевірили правильність виконання алгоритму та програми.