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 3

Виконав студент ІП-15, Борисик Владислав Тарасович
(шифр, прізвище, ім'я, по батькові)

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

Київ 2021

Лабораторна робота №7
Дослідження лінійного пошуку в послідовностях

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

Варіант 3

Задача

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

1. Опису трьох змінних індексованого типу з 10 символьних значень.
2. Ініціювання двох змінних виразами згідно з варіантом (рис.).
3. Ініціювання третьої змінної рівними значеннями двох попередніх змінних.
4. Обробки третьої змінної згідно з варіантом.

3	55 - 2 * i	40 + 3 * i	Знайти значення, яке є мінімальним та, більшим за задане P

Постановка задачі

За умовою задачі потрібно створити три змінні індексованого типу (масиви) з 10 символьних значень, ініціювати їх згідно виразами, створити третю змінну індексованого типу і знайти в ній значення індексу, яке є мінімальним та більшим за задане P.

Результатом розв'язку є значення, яке є мінімальним та більшим за задане P у третій змінній індексованого типу.

Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Ім’я	Призначення
Задане P	Цілий	P	Початкове дане
Перший масив	Символьний	array1	Проміжне дане
Другий масив	Символьний	array2	Проміжне дане
Третій масив	Символьний	result_array	Проміжне дане, результат
Мінімальне та більше	Цілий	minimal_value	Результат

значення за P			

Для операції порівняння будемо використовувати оператор == .
Для того, щоб показати, що параметр функції передається за посиланням будемо використовувати оператор &.

Для ініціалізації елементів першого масиву згідно з формулою (55 - 2 * i) будемо використовувати власну функцію init_first_array(array). Ця функція приймає 1 аргумент:
array - масив символьного типу,
Ця функція повертає масив символьного типу, елементи якого, ініціалізовані згідно з формулою.

Для ініціалізації елементів другого масиву згідно з формулою (40 + 3 * i) будемо використовувати власну функцію init_second_array(array). Ця функція приймає 1 аргумент:
array - масив символьного типу,
Ця функція повертає масив символьного типу, елементи якого, ініціалізовані згідно з формулою.

Для ініціалізації елементів третього масиву символами з кодом 0 будемо використовувати власну функцію init_result_array(array). Ця функція приймає 1 аргумент:
array - масив символьного типу,
Ця функція повертає масив символьного типу, елементи якого, ініціалізовані символами з кодом 0.

Для знаходження спільних елементів двох масивів будемо використовувати власну функцію find_similar(array1, array2, &result_array). Ця функція приймає 3 аргументи:
array1 - перший масив символьного типу,
array2 - другий масив символьного типу,
result_array - третій масив символьного типу, куди будуть записані спільні елементи з array1 і array2.
Ця функція записує елементи символьного типу, які є спільними для array1 і array2, в масив символьного типу result_array. Не повертає нічого.

Для знаходження значення, яке є мінімальним та більшим за задане P будемо використовувати власну функцію find_minimal(array, p). Ця функція приймає 2 аргументи:
array - масив символьного типу,
p - ціле число. Задане користувачем P.
Ця функція повертає символ, який є мінімальним та більшим за задане P.

1) Створюємо перший масив array1.

2) За допомогою арифметичного циклу заповнюємо array1 символами, згідно з

формулою
55 - 2 * i
3) Створюємо другий масив array2.

4) За допомогою арифметичного циклу заповнюємо array2 символами, згідно з формулою
40 + 3 * i
5) Створюємо третій масив result_array.

6) За допомогою арифметичного циклу заповнюємо його символами з кодом 0

7) За допомогою власної функції find_similar(array1, array2, &result_array), знаходимо спільні елементи з array1 і array2.

8) Виводимо масив result_array.

9) Створюємо змінну P цілочисельного типу

10) Просимо користувача ввести число P

11) Створюєму змінну minimal_value символьного типу і за допомогою власної функції find_minimal(array, p) присвоюємо результат роботи функції змінній minimal_value.

12) Виводимо змінну minimal_value.

Розв'язання

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

Крок 1. Визначимо основні дії

Крок 2. Створення першого масиву array1 і його ініціалізація функцією init_first_array(array).

Крок 3. Створення другого масиву array2 і його ініціалізація функцією init_second_array(array).

Крок 4. Створення третього масиву result_array і його ініціалізація функцією init_result_array(array).

Крок 5. Знаходимо спільні елементи з array1 і array2.

Крок 6. Виведення масиву result_array

Крок 7. Створення змінної P цілочисельного типу

Крок 8. Введення P

Крок 9. Створення змінної minimal_value змінного типу і присвоєння їй результат роботи функції find_minimal(array, p).

Крок 10. Виведення minimal_value.

Псевдокод

Підпрограма init_first_array(array):

Початок

повторити

для i **від** 0 **до** 10

    array[i] := 55 - 2 * i

все повторити

повернути array

Кінець

Підпрограма init_second_array(array):

Початок

повторити

для i **від** 0 **до** 10

    array[i] := 40 + 3 * i

все повторити

повернути array

Кінець

Підпрограма init_result_array(array):

Початок

повторити

для i **від** 0 **до** 10

```
    array[i] := 0
все повторити
повернути array
Кінець


Підпрограма find_similar(array1, array2, &result_array):
Початок
повторити
для i від 0 до 10
    повторити
        для j від 0 до 10
            якщо array1[i] == array2[j]
                то
                    result_array[i] := array2[j]
            все якщо
    все повторити
все повторити
Кінець

Підпрограма find_minimal(array, p):
Початок
повторити
для i від 9 до 0
    якщо array[i] > p
        то
            повернути array[i]
    все якщо
все повторити
Кінець

Основна програма:
```

Початок

array1[10] := init_first_array(array1)

array2[10] := init_second_array(array2)

result_array[10] := init_result_array(result_array)

find_similar(array1, array2, &result_array)

Виведення result_array

P := 0

Введення P

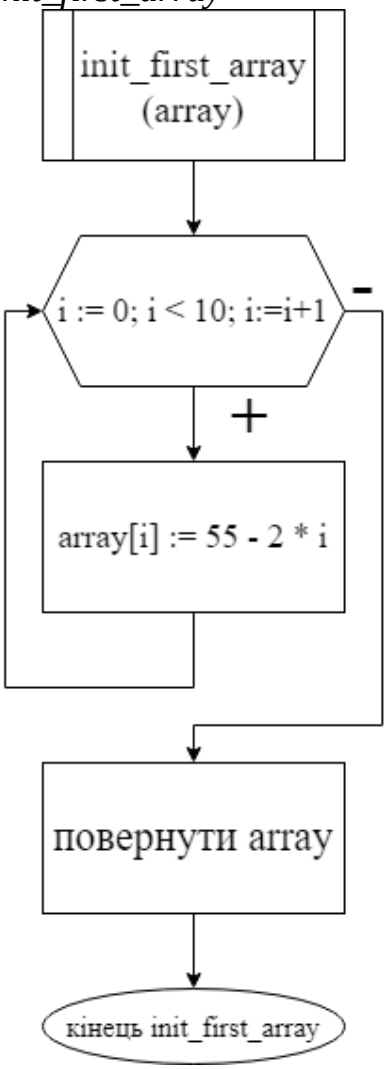
minimal_value = find_minimal(result_array, P)

Виведення minimal_value

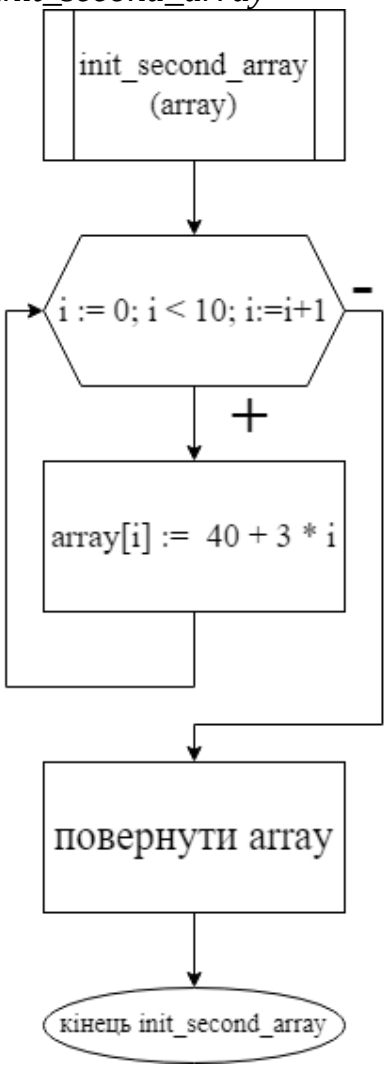
Кінець

Блок-схема алгоритму

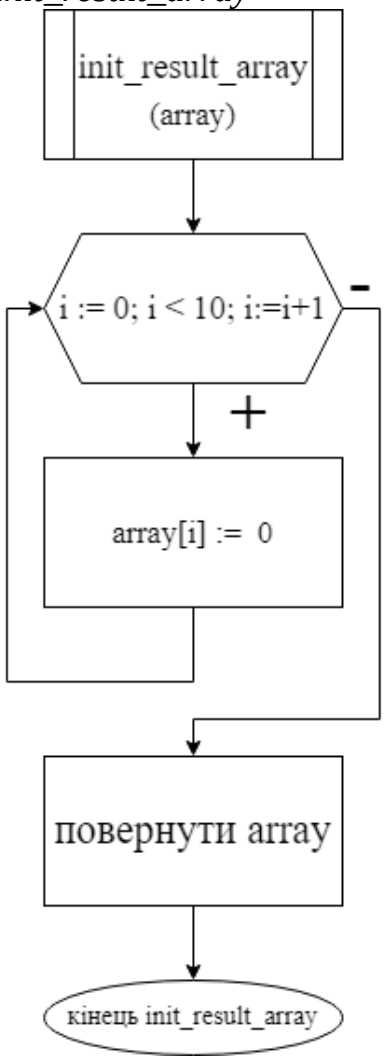
Підпрограма init_first_array



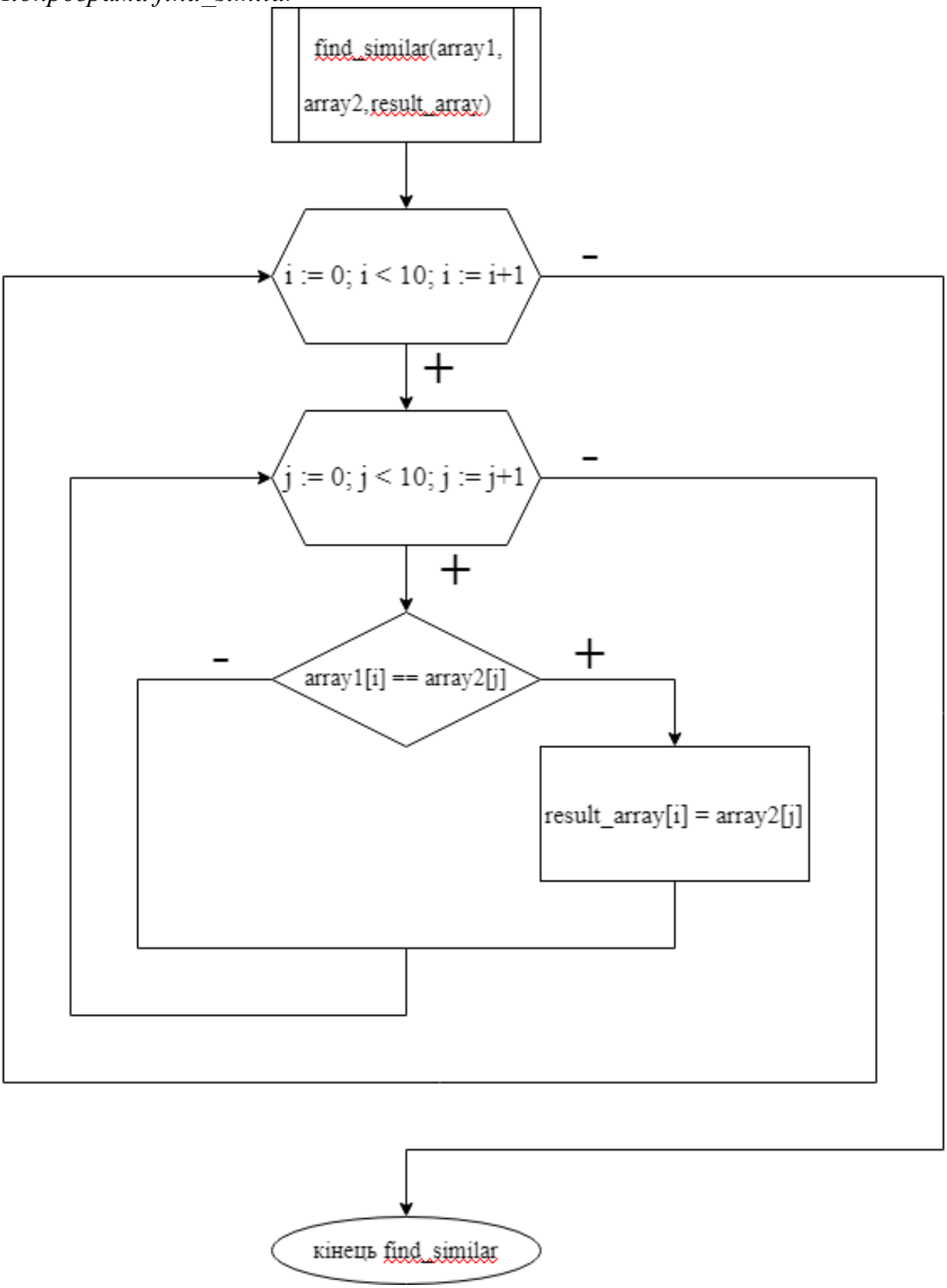
Підпрограма init_second_array



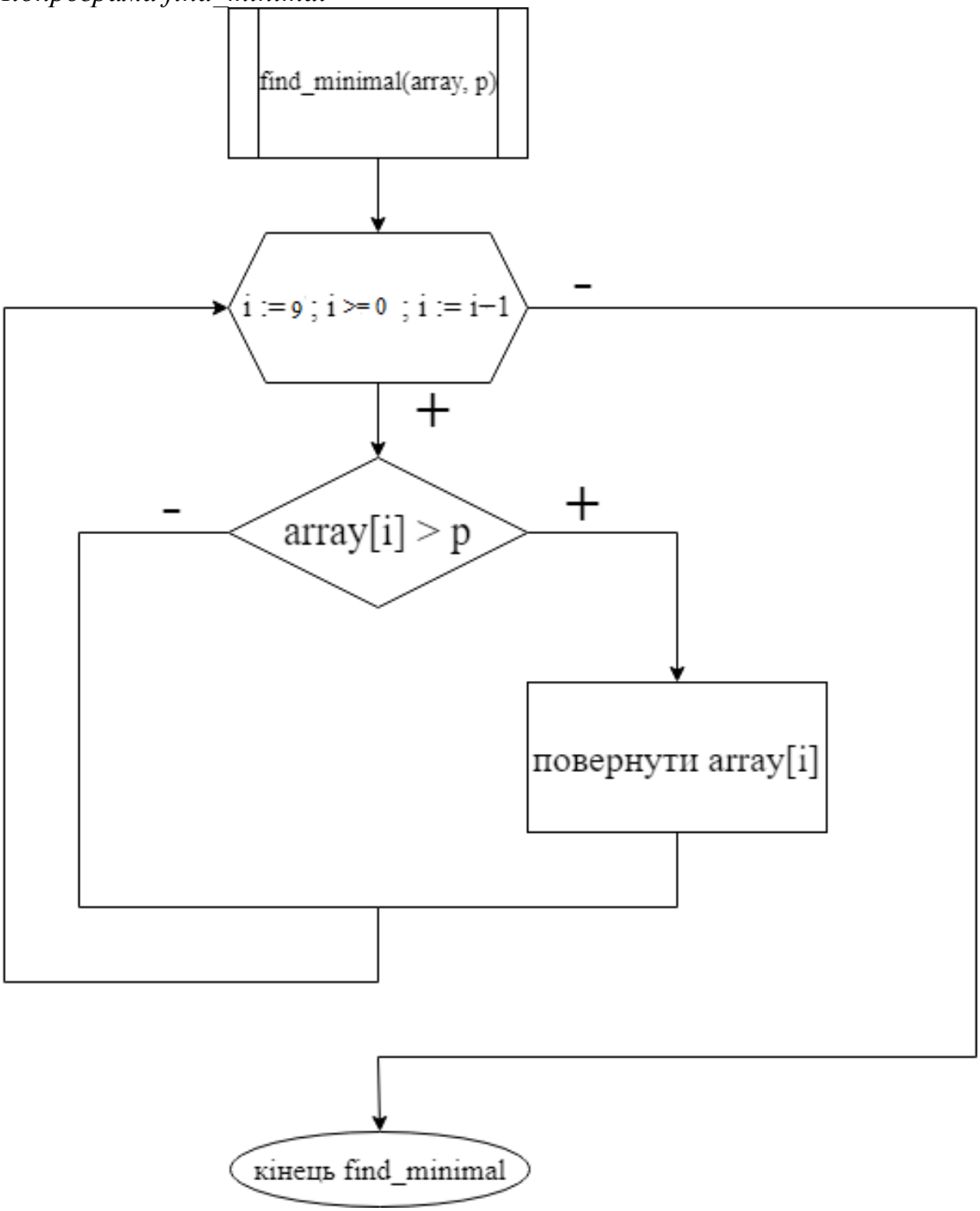
Підпрограма init_result_array



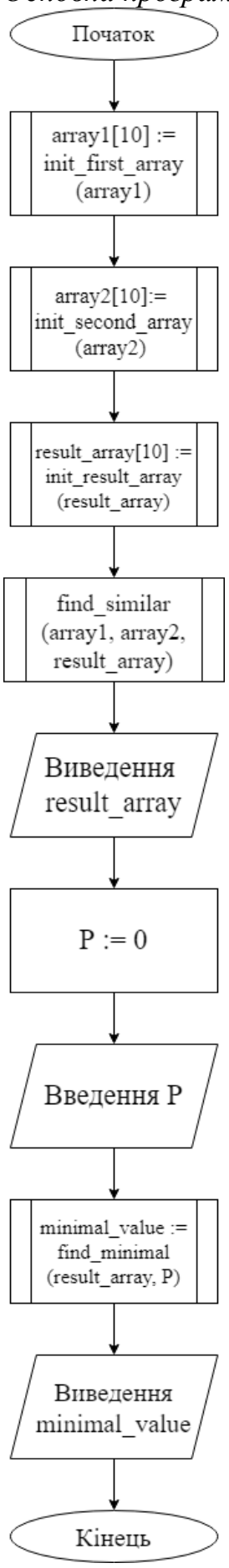
Підпрограма find_similar



Підпрограма find_minimal



Основна програма



Код

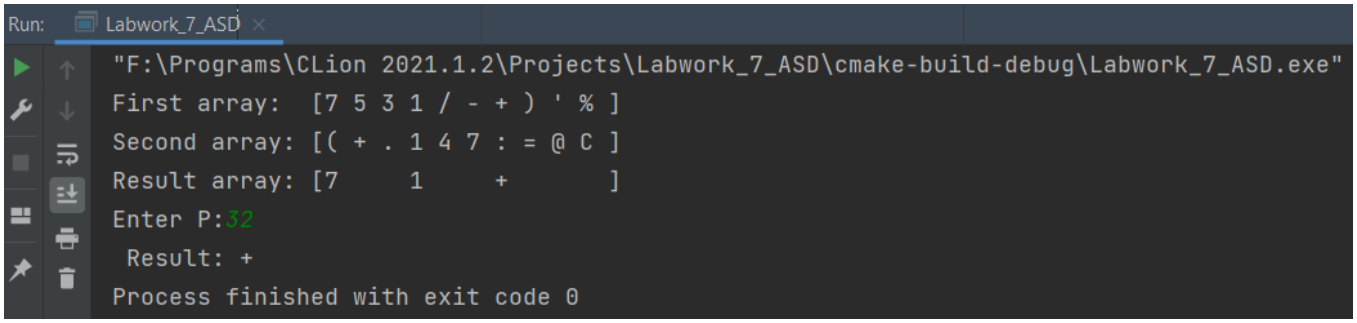
```cpp
#include <iostream>
using namespace std;

/** Выводить в консоль елементи масиву.
 *  Нічого не повертає
 *  array:      масив
 *  array_size: довжина масиву */
void print_array(char array[], int array_size);

/** Ініціалізує перший масив згідно з формулою .
 *  Нічого не повертає
 *  array: масив */
void init_first_array(char (&array)[10]);

/** Ініціалізує другий масив згідно з формулою.
 *  Нічого не повертає
 *  array: масив */
void init_second_array(char (&array)[10]);

/** Ініціалізує третій масив згідно з формулою.
 *  Нічого не повертає
 *  array: масив */
void init_result_array(char (&array)[10]);

/** Знаходить спільні елементи в двох масивах
 *  Нічого не повертає
 *  array1:       перший масив
```

```
 *  array2:       другий масив
 *  result_array: результуючий масив (всі дані зберігаються в ньому, так как ми передаємо його через посилання)
void find_similar(const char array1[], const char array2[], char (&result_array)[10]);

/** Знаходить значення, яке є мінімальним та, більшим за задане P
 *  Повертає значення, яке є мінімальним та, більшим за задане P
 *  array: масив
 *  p:     задане число */
char find_minimal(const char array[], int p);


int main() {
    // створюємо перший масив
    char array1[10];
    // заповнюємо його символами, згідно з формулою
    init_first_array( &: array1);

    printf("First array:  ");
    print_array(array1, array_size: 10);

    // створюємо другий масив
    char array2[10];
    // заповнюємо його символами, згідно з формулою
    init_second_array( &: array2);

    printf("Second array: ");
    printf("Second array: ");
    print_array(array2, array_size: 10);

    // створюємо результуючий масив
    char result_array[10];
    // заповнюємо його символами з кодом 0
    init_result_array( &: result_array);

    // знаходимо спільні елементи перших двох масивів і записуємо їх у третій
    find_similar(array1, array2, &: result_array);

    printf("Result array: ");
    print_array(result_array, array_size: 10);

    int P;
    printf("Enter P: ");
    cin >> P;

    char minimal_value = find_minimal(result_array, P);
    printf("Result: %c", minimal_value);
}


void print_array(char array[], int array_size){
    cout << "[";

    // ітеруємось через кожен елемент масиву і виводимо їх в консоль
```

```
79          // ітеруємось через кожен елемент масиву і виводимо їх в консоль
80          for(int i = 0; i < array_size; i++)
81              cout << array[i] << " ";
82
83          cout << "]" << endl;
84      }
85
86
87      void init_first_array(char (&array)[10]){
88          for(int i = 0; i < 10; i++)
89              array[i] = static_cast<char>(55 - 2 * i);
90      }
91
92
93      void init_second_array(char (&array)[10]){
94          for(int i = 0; i < 10; i++)
95              array[i] = static_cast<char>(40 + 3 * i);
96      }
97
98
99      void init_result_array(char (&array)[10]){
100         for(int i = 0; i < 10; i++)
101             array[i] = 0;
102     }
103
104
```

```
105  void find_similar(const char array1[], const char array2[], char (&result_array)[10]){
106      // ітеруємось через елементи першого масиву
107      for(int i = 0; i < 10; i++)
108          // ітеруємось через елементи другого масиву
109          for(int j = 0; j < 10; j++)
110              // якщо елемент і першого масиву співпав х елементом j другого масиву
111              if(array1[i] == array2[j])
112                  // записуємо цей елемент в результуючий масив
113                  result_array[i] = array2[j];
114  }
115
116
117  char find_minimal(const char array[], int p){
118      // ітеруємось через елементи  масиву
119      for(int i=9; i >= 0; i--)
120          // якщо символ більший за задане P
121          if(array[i] > p)
122              // повертаєм цей символ
123              return array[i];
124      return 0;
125  }
126
```

Результат виконання програми



Випробування алгоритму

Блок	Дія
	Початок
1	array1[10] = []
2	array1[10] = [7 5 3 1 / - +) ' %]
3	array2[10] = []
4	array2[10] = [(+ . 1 4 7 : = @ C]
5	result_array[10] = []
6	result_array[10] = [ ]
7	find_similar(array1, array2, &result_array)
8	Виведення result_array[10] = [7 1 +]
9	Введення P = 32
10	minimal_value = find_minimal(result_array, p)
11	Виведення minimal_value = +
12	Кінець

Висновок

Протягом сьомої лабораторної роботи я дослідив методи послідовного пошуку у впорядкованих і невпорядкованих послідовностях та набув практичних навичок їх використання під час складання програмних специфікацій.