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

«Алгоритми пошуку в послідовностях»

Варіант 13

Виконав студент ІП-43 Дутов Іван Андрійович

Перевірила Вітковська Ірина Іванівна

Київ 2024

Лабораторна робота №6
АЛГОРИТМИ ПОШУКУ В ПОСЛІДОВНОСТЯХ

Задача. Ініціалізувати два символьні масиви довжиною 10, знайти суму кодів мінімального та максимального елементів серед спільних елементів цих двох масивів.

Постановка задачі

Для ініціалізації двох символьних масивів використаємо підпрограму `initialize_array`, що ітеративно генеруватиме необідні нам послідовності $62 + 3i$ та $74 - i$ для $i \in [0, 9]$

Через те що перший масив впорядкований за зростанням, можна застосувати бінарний пошук символа у цьому масиві завдяки підпрограмі `binary_search`. Також застосовуємо лінійний пошук через підпрограму `linear_search`. `find_common_elements` використовується для знаходження спільних елементів двох масивів та збереження їх у інший масив. Ця підпрограма використовує арифметичний цикл за кожним символом другого масиву та згодом знаходить цей символ в іншому масиві за допомогою заданого алгоритму пошуку.

Знаходимо мінімальний та максимальний елемент масиву спільних символів для першого та другого масивів за допомогою підпрограми `find_min_max`, що використовує лінійний пошук, ітеративно зберігаючи коди символів, менші за мінімальний або більший за максимальний поки що знайдені коди.

Табл. 1: Таблиця імен змінних

Програма			
Змінна	**Тип**	**Ім'я**	**Призначення**
Розмір початкових масивів	символ	`ARR_SIZE`	Константа
Перший масив	масив символів	`arr1`	Змінна
Другий масив	масив символів	`arr2`	Змінна
Мінімальний спільний код	символ	`min`	Змінна
Максимальний спільний код	символ	`max`	Змінна
Розмір масиву спільних символів	ціле	`common_item_count`	Змінна
Сума мінімального та максимального кодів для бінарного пошуку	ціле	`binary_result`	Змінна
Сума мінімального та максимального кодів для лінійного пошуку	ціле	`linear_result`	Змінна
Підпрограма `initialize_array`			
Масив для символьної ініціалізації	масив символів	`arr`	Формальний параметр
Розмір масиву	ціле	`size`	Формальний параметр
Початковий код символу	ціле	`start`	Формальний параметр
Крок ходи ініціалізації	ціле	`step`	Формальний параметр
Підпрограма `binary_search`			
Впорядкований масив для пошуку заданого символа	масив символів	`arr`	Формальний параметр

Продовження на наступній сторінці...

Програма (продовження)			
Змінна	**Тип**	**Ім'я**	**Призначення**
Розмір впорядкованого масиву	ціле	`size`	Формальний параметр
Шуканий символ	символ	`target`	Формальний параметр
Індекс найменшого елемента для пошуку	ціле	`low`	Змінна
Індекс найбільшого елемента для пошуку	ціле	high	Змінна
Індекс можливого шуканого елемента	ціле	`mid`	Змінна
Підпрограма `linear_search`			
Впорядкований масив для пошуку заданого символа	масив символів	`arr`	Формальний параметр
Розмір впорядкованого масиву	ціле	`size`	Формальний параметр
Шуканий символ	символ	`target`	Формальний параметр
Підпрограма `find_common_elements`			
Перший (відсортований) масив для пошуку	масив символів	`search_arr`	Формальний параметр
Другий символьний масив для пошуку спільних елементів	масив символів	`arr`	Формальний параметр
Розмір, спільний для масивів	ціле	`size`	Формальний параметр
Масив для збереження спільних символів	масив символів	`common`	Формальний параметр
Функція пошуку	вказівник на функцію	`search`	Формальний параметр
Число спільних елементів	ціле	`common_count`	Змінна
Підпрограма `find_min_max`			
Масив символів для пошуку мінімального та максимального значень	масив символів	`arr`	Формальний параметр
Розмір масиву	ціле	`size`	Формальний параметр
Мінімальний елемент масиву	символ	`*min`	Формальний параметр
Максимальний елемент масиву	символ	`*max`	Формальний параметр
Поточний елемент масиву під час ітерації	символ	`arr[i]`	Змінна

Крок 1. Визначимо основні дії.

Крок 2. Уточнимо дії ініціалізації першого та другого масивів.

Крок 3. Уточнимо дії знаходження спільних елементів двома методами.

Крок 4. Уточнимо дії обчислення суми кодів максимального та мінімального спільних елементів.

Крок 5. Уточнимо дію перевірки правильності алгоритму

Псевдокод програми

Крок 1

1. **Початок**.
2. Ініціалізація необхідних змінних
3. Знаходження спільних елементів за допомогою бінарного пошуку.
4. Обчислення суми кодів у випадку бінарного пошуку.
5. Знаходження спільних елементів за допомогою лінійного пошуку.
6. Обчислення суми кодів у випадку лінійного пошуку.
7. Перевірка правильності алгоритму.
8. **Кінець**.

Крок 2

1. **Початок**.
2. Ініціалізація необхідних змінних
   2.1 `ARR_SIZE := 10`
   2.2 Створення змінних `arr1, arr2, common, min, max`
   2.3 `initialize_array(arr1, ARR_SIZE, 62, 3)`
   2.4 `initialize_array(arr2, ARR_SIZE, 74, -1)`
3. Знаходження спільних елементів за допомогою бінарного пошуку.
4. Обчислення суми кодів у випадку бінарного пошуку.
5. Знаходження спільних елементів за допомогою лінійного пошуку.
6. Обчислення суми кодів у випадку лінійного пошуку.
7. Перевірка правильності алгоритму.
8. **Кінець**.

Крок 3

1. **Початок**.
2. Ініціалізація необхідних змінних
   2.1 `ARR_SIZE := 10`
   2.2 Створення змінних `arr1, arr2, common, min, max`
   2.3 `initialize_array(arr1, ARR_SIZE, 62, 3)`
   2.4 `initialize_array(arr2, ARR_SIZE, 74, -1)`
3. `common_count := find_common_elements(arr1, arr2, ARR_SIZE, common, binary_search)`
4. Обчислення суми кодів у випадку бінарного пошуку.

5. `common_count := find_common_elements(arr1, arr2, ARR_SIZE, common, linear_search)`
6. Обчислення суми кодів у випадку лінійного пошуку.
7. Перевірка правильності алгоритму.
8. **Кінець**.

Крок 4

1. **Початок**.
2. Ініціалізація необхідних змінних
   2.1 `ARR_SIZE := 10`
   2.2 Створення змінних `arr1, arr2, common, min, max`
   2.3 `initialize_array(arr1, ARR_SIZE, 62, 3)`
   2.4 `initialize_array(arr2, ARR_SIZE, 74, -1)`
3. `common_count := find_common_elements(arr1, arr2, ARR_SIZE, common, binary_search)`
4. Обчислення суми кодів у випадку бінарного пошуку.
   4.1 `find_min_max(common, common_count, &min, &max)`
   4.2 `binary_result := min + max`
5. `common_count := find_common_elements(arr1, arr2, ARR_SIZE, common, linear_search)`
6. Обчислення суми кодів у випадку лінійного пошуку.
   6.1 `find_min_max(common, common_count, &min, &max)`
   6.2 `linear_result := min + max`
7. Перевірка правильності алгоритму.
8. **Кінець**.

Крок 5

1. **Початок**.
2. Ініціалізація необхідних змінних
   2.1 `ARR_SIZE := 10`
   2.2 Створення змінних `arr1, arr2, common, min, max`
   2.3 `initialize_array(arr1, ARR_SIZE, 62, 3)`
   2.4 `initialize_array(arr2, ARR_SIZE, 74, -1)`
3. `common_count := find_common_elements(arr1, arr2, ARR_SIZE, common, binary_search)`
4. Обчислення суми кодів у випадку бінарного пошуку.
   4.1 `find_min_max(common, common_count, &min, &max)`
   4.2 `binary_result := min + max`

5. `common_count := find_common_elements(arr1, arr2, ARR_SIZE, common, linear_search)`
6. Обчислення суми кодів у випадку лінійного пошуку.
   6.1 `find_min_max(common, common_count, &min, &max)`
   6.2 `linear_result := min + max`
7. **Якщо** `binary_result != linear_result`,
   то
   7.1 **Вивід** «ПОМИЛКА! Відповіді не співпали!»
8. **Кінець**.

Псевдокод підпрограми `initialize_array`

1. **Початок** `initialize_array(arr, size, start, step)`
2. **Повторити для** `i від 1 до size`
   2.1 `arr[i] := start + step * i`
   все повторити
3. **Кінець**

Псевдокод підпрограми `binary_search`

1. **Початок** `binary_search(arr, size, target)`
2. `low := 0`
3. `high := size - 1`
4. **Поки** `low <= high`
   4.1 `mid := low + (high - low) / 2`
   4.2 **Якщо** `arr[mid] > target`,
   то
   4.2.1 `high := mid - 1`
   інакше
   4.3 **Якщо** `arr[mid] < target`,
   то
   4.3.1 `low := mid + 1`
   інакше
   4.4 `return true`
   все якщо
   все поки
5. `return false`
6. **Кінець**

Псевдокод підпрограми `linear_search`

1. **Початок** `binary_search(arr, size, target)`
2. **Повторити для** `від 0 до size - 1`

   2.1 **Якщо** `arr[i] == target`,
   то

   2.1.1 `return true`

   все повторити
3. `return false`
4. **Кінець**

Псевдокод підпрограми `find_common_elements`

1. **Початок** `find_common_elements(search_arr, arr, size, common, search)`
2. **Повторити для** `i від 0 до size - 1`

   2.1 **Якщо** `search(search_arr, size, arr[i])`,
   то

   2.1.1 `common[common_count] = arr[i]`

   2.1.2 `common_count := common_count + 1`

   все повторити
3. **Кінець**

Псевдокод підпрограми `find_min_max`

1. **Початок** `find_min_max(arr, size, *min, *max)`
2. `*min := arr[0]`
3. `*max := arr[0]`
4. **Повторити для** `i від 1 до size - 1`

   4.1 **Якщо** `arr[i] < *min`,
   то

   4.1.1 `*min := arr[i]`

   інакше

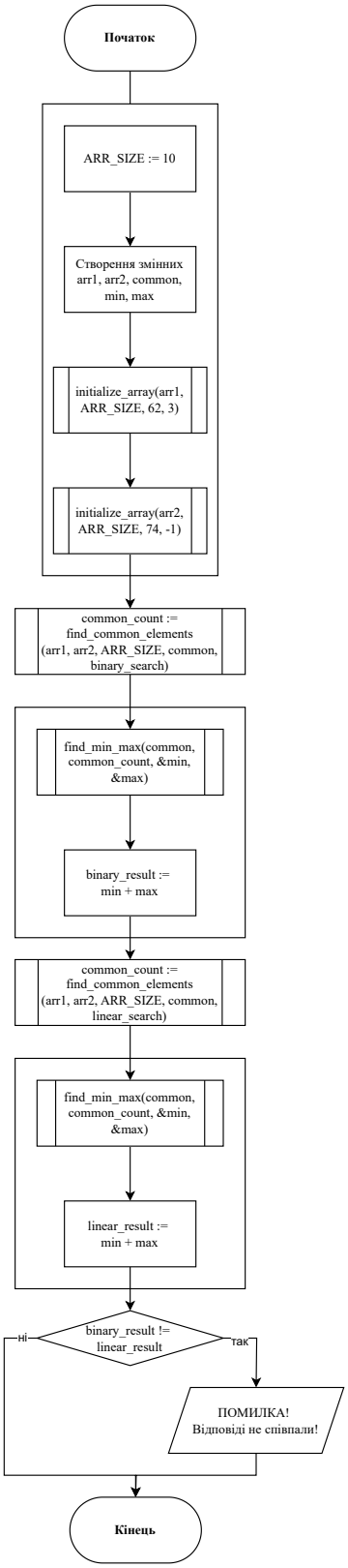
   4.2 **Якщо** `arr[i] > *max`,
   то

   4.2.1 `*max := arr[i]`

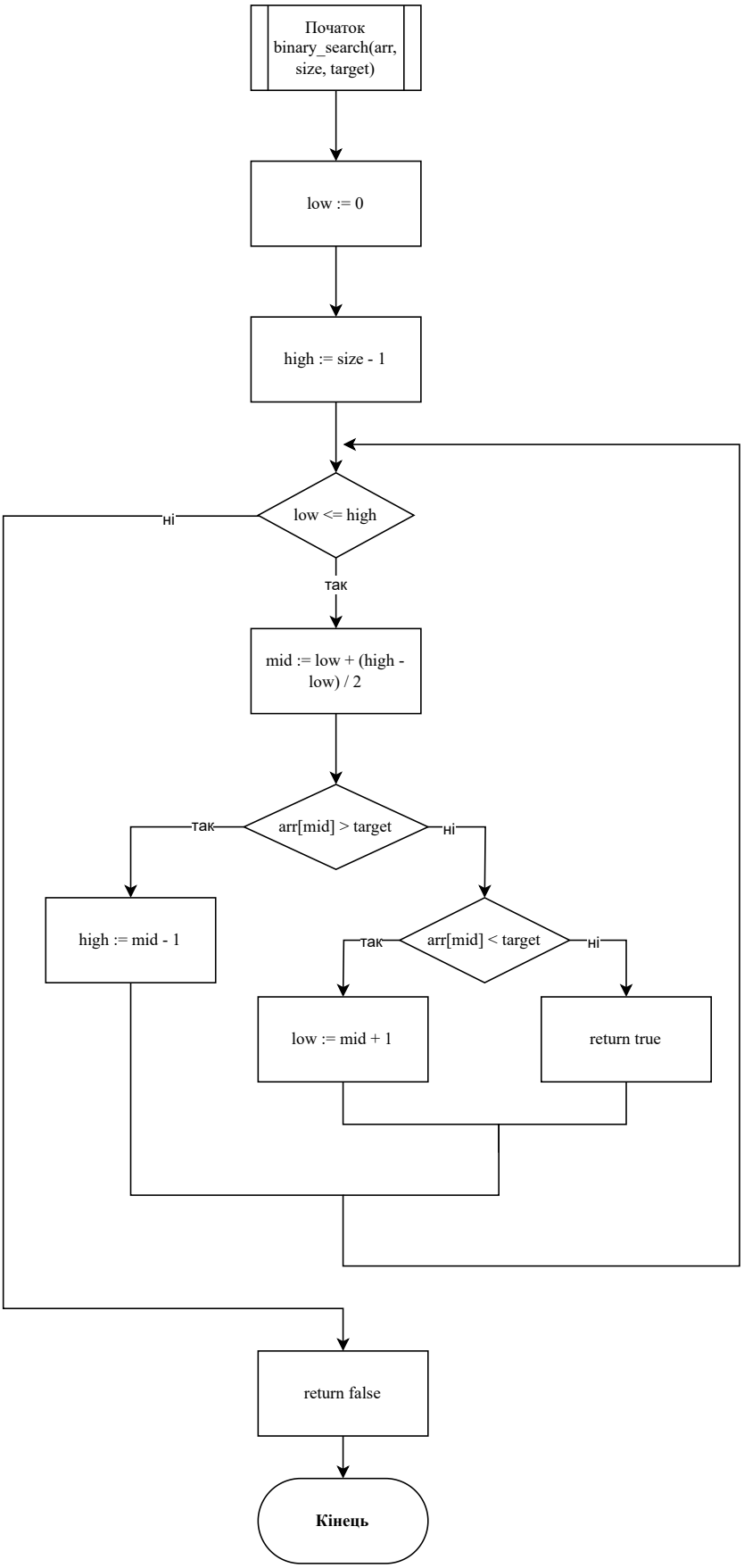
   все якщо

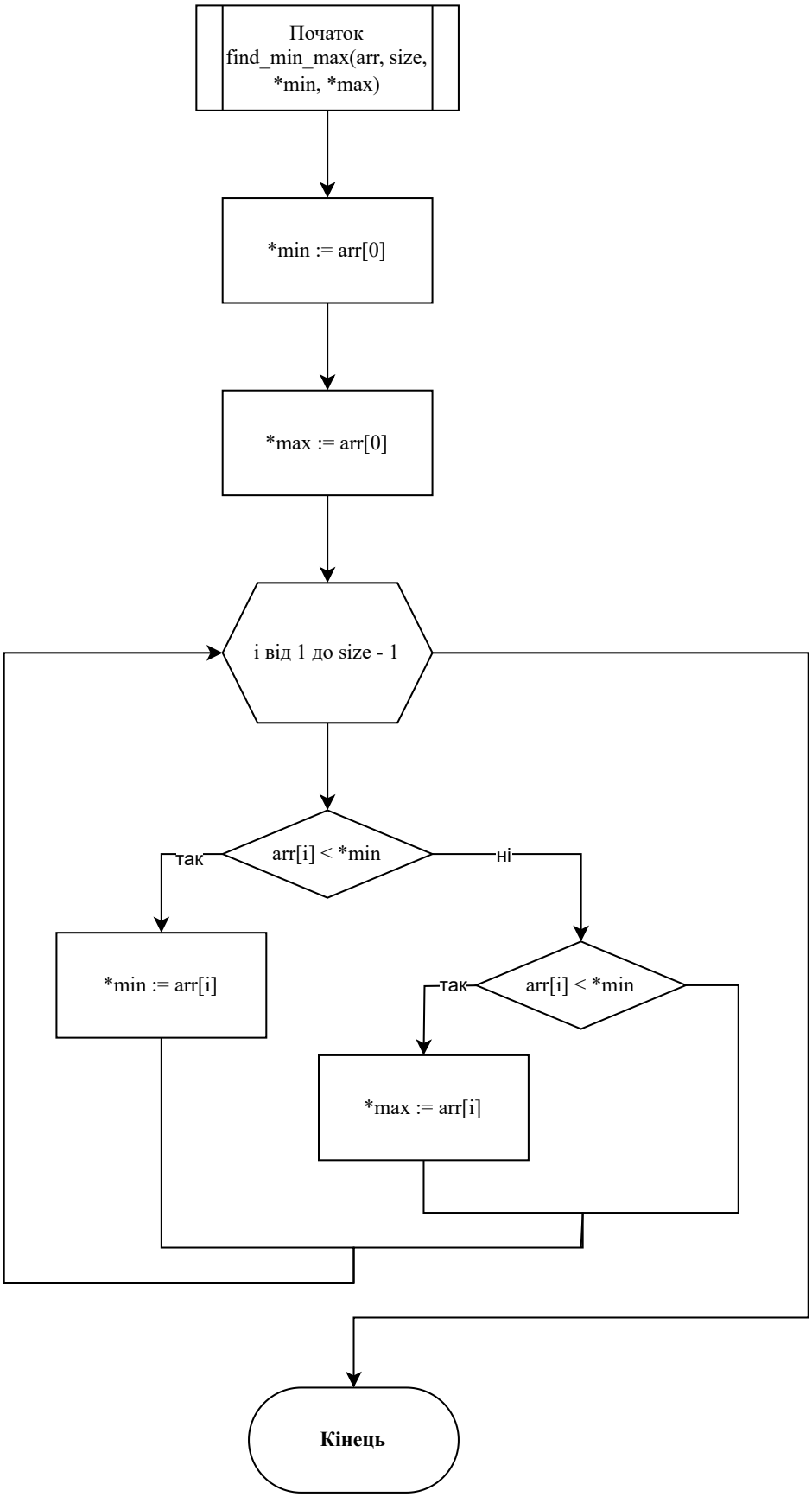
   все повторити
5. **Кінець**

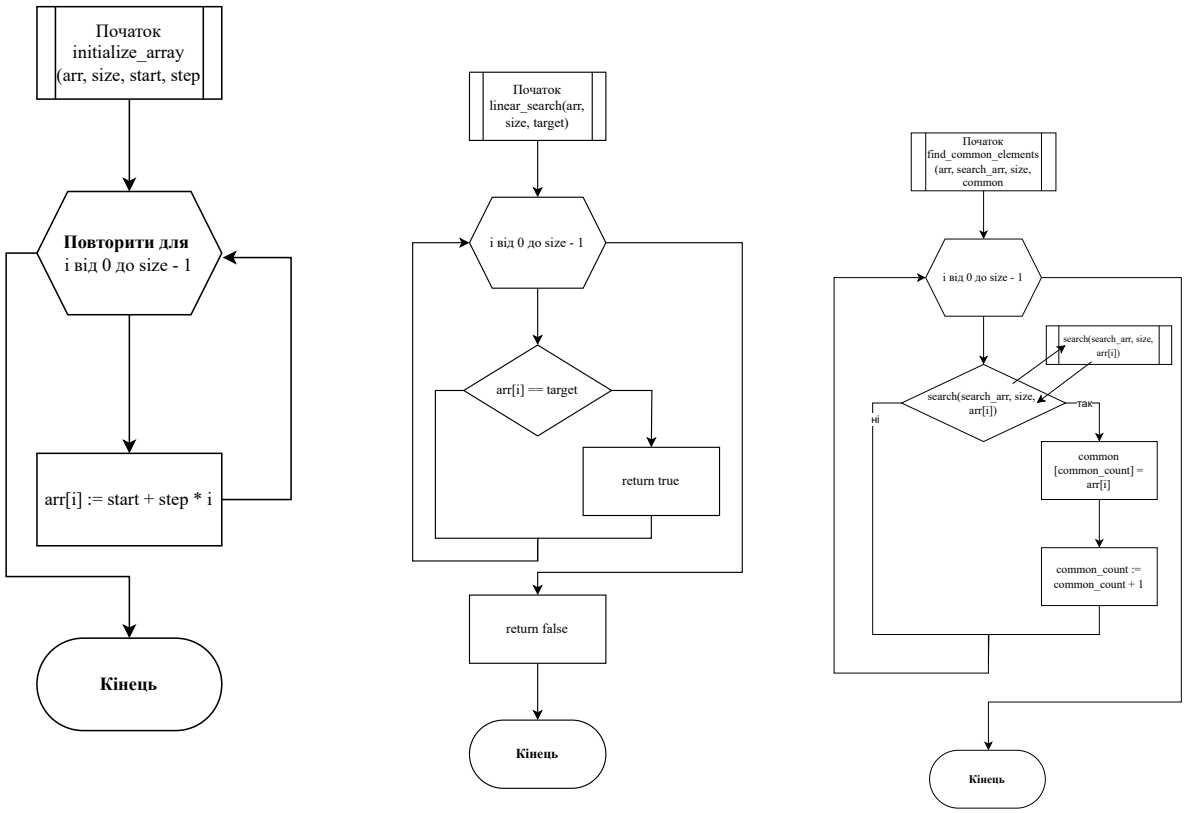
Блок-схема алгоритму



Блок-схеми допоміжних алгоритів




Початок
find_min_max(arr, size, *min, *max)
*min := arr[0]
*max := arr[0]
i від 1 до size - 1
arr[i] < *min
так
ні
*min := arr[i]
arr[i] < *min
так
*max := arr[i]
Кінець



Перевірка правильності алгоритмів

Перший масив `arr1`	`">ADGJMPSVY"`
Другий масив `arr2`	`"JIHGFEDCBA"`
Масив спільних елементів після бінарного пошуку `common`	`"JGDA"`
Сума мінімального та максимального спільних елементів при бінарному пошуку	`'J' + 'A' = 139`
Масив спільних елементів після лінійного пошуку `common`	`"JGDA"`
Сума мінімального та максимального спільних елементів при лінійного пошуку	`'J' + 'A' = 139`

Код програми мовою C

```c
#include <stdbool.h>
#include <stddef.h>
#include <stdio.h>

#define ARR_SIZE 10
typedef bool (*Search)(const char arr[], size_t size, char target);

void initialize_array(char arr[], size_t size, int start, int step);
bool binary_search(const char arr[], size_t size, char target);
bool linear_search(const char arr[], size_t size, char target);
int find_common_elements(const char search_arr[], const char arr[], size_t size,
```

```
                    char common[], Search search);
void find_min_max(const char arr[], size_t size, char *min, char *max);

int main() {
  char arr1[ARR_SIZE], arr2[ARR_SIZE], common[ARR_SIZE], min, max;

  initialize_array(arr1, ARR_SIZE, 62, 3);
  initialize_array(arr2, ARR_SIZE, 74, -1);

  int common_count =
      find_common_elements(arr1, arr2, ARR_SIZE, common, binary_search);
  find_min_max(common, common_count, &min, &max);
  int binary_result = (int)min + (int)max;

  common_count =
      find_common_elements(arr1, arr2, ARR_SIZE, common, linear_search);
  find_min_max(common, common_count, &min, &max);
  int linear_result = (int)min + (int)max;

  if (binary_result != linear_result) {
    printf("ПОМИЛКА! Відповіді не співпали!");
  }
  return 0;
}

void initialize_array(char arr[], size_t size, int start, int step) {
  for (size_t i = 0; i <= size - 1; i++) {
    arr[i] = (char)(start + step * i);
  }
}

bool binary_search(const char arr[], size_t size, char target) {
  int low = 0;
  int high = size - 1;

  while (low <= high) {
    int mid = low + (high - low) / 2;

    if (arr[mid] > target) {
      high = mid - 1;
    } else if (arr[mid] < target) {
      low = mid + 1;
    } else {
      return true;
    }
  }
  return false;
}

bool linear_search(const char arr[], size_t size, char target) {
  for (int i = 0; i < size; i++) {
    if (arr[i] == target) {
      return true;
    }
  }
  return false;
}

int find_common_elements(const char search_arr[], const char arr[], size_t size,
                    char common[], Search search) {
```

```
  int common_count = 0;

  for (size_t i = 0; i <= size - 1; i++) {
    if (search(search_arr, size, arr[i])) {
      common[common_count++] = arr[i];
    }
  }

  return common_count;
}

void find_min_max(const char arr[], size_t size, char *min, char *max) {
  *min = arr[0];
  *max = arr[0];

  for (size_t i = 1; i <= size - 1; i++) {
    if (arr[i] < *min) {
      *min = arr[i];
    } else if (arr[i] > *max) {
      *max = arr[i];
    }
  }
}
```

Висновок

Ми дослідили методи пошуку у впорядкованих та невпорядкованих послідовностях та набули практичних навичок їх використання під час складання програмної специфікації з підпрограмами знаходження спільних символів у двох символьних масивах та знаходження мінімального та максимального кодів у символьному масиві.