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»

«Лінійні алгоритми»

Варіант 1

Виконав студент ІП-45 Янов Богдан Євгенійович

Перевірила старший викладач Вечерковська Анастасія Сергіївна

Київ 2024

Лабораторна робота №6

Тема - алгоритми пошуку в послідовностях**.**
Мета – дослідити методи пошуку у впорядкованих і невпорядкованих послідовностях та набути практичних навичок їх використання під час складання програмних специфікацій.

Завдання

Знайти рівні елементи серед двох масивів
Знайти два максимальних елементи та їх суму

Постановка задач

1. Визначимо основні дії.
2. Деталізуємо дію ініціалізації початкових масивів
3. Деталізуємо дію виводу кота на екран
4. Деталізуємо дію знаходження масиву рівних елементів
5. Деталізуємо дію знаходження двох максимальних елементів та їх суми

Математична модель

Name	**Type**	**Purpose**
first_array	unsigned char[]	Initial data
second_array	unsigned char[]	Initial data
common_array	unsigned char[]	Array of common elements
SIZE	#define	Size of arrays
common_count	int	Count of common elements
first_largest, second_largest	unsigned char	Largest elements of common array
sum	unsigned char	Sum of largest elements

SIZE = 10
Expression for calculating the element of the 1st array: 117 + i
Expression for calculating the element of the 2nd array: 127 – 2 * i

Псевдокод

Крок 1

Begin

Initialize initial arrays

Print cat

Find common values

Find two largest and sum

End

Крок 2

Begin

unsigned char first_array[SIZE]

unsigned char second_array[SIZE]

unsigned char common_array[SIZE]

unsigned char first_largest

unsigned char second_largest

generate_first_array(first_array, SIZE)

generate_second_array(second_array, SIZE)

Print cat

Find common values

Find two largest and sum

End

Крок 3

Begin

unsigned char first_array[SIZE]

unsigned char second_array[SIZE]

unsigned char common_array[SIZE]

unsigned char first_largest

unsigned char second_largest

generate_first_array(first_array, SIZE)

generate_second_array(second_array, SIZE)

print_cat()

Find common values

Find two largest and sum

End

Крок 4

Begin

unsigned char first_array[SIZE]
unsigned char second_array[SIZE]
unsigned char common_array[SIZE]
unsigned char first_largest
unsigned char second_largest

generate_first_array(first_array, SIZE)
generate_second_array(second_array, SIZE)

print_cat()

common_count = find_common_values(first_array, second_array, common_array)
Find two largest and sum

End

Крок 5

Begin

unsigned char first_array[SIZE]
unsigned char second_array[SIZE]
unsigned char common_array[SIZE]
unsigned char first_largest
unsigned char second_largest

generate_first_array(first_array, SIZE)
generate_second_array(second_array, SIZE)

print_cat()

common_count = find_common_values(first_array, second_array, common_array)
find_two_largest(common_array, common_count, &first_largest, &second_largest)
sum = first_largest + second_largest

Output first_largest
Output second_largest
Output sum

End

Допоміжні алгоритми

```
Begin linear_search(array, target)
   Loop for i from 0 to SIZE
      If array[i] == target
         Return i
      End if
   End loop

   Return -1
End

Begin binary_search(array, target)
   left = 0
   right = SIZE - 1

   While left <= right
      mid = left + (right - left) / 2

      If arr[mid] == target
         Return mid
      Else if arr[mid] < target
         right = mid - 1
      Else
         left = mid + 1
      End if
   End while

   Return -1
End

Begin find_common_values(first_array, second_array, common_array)
   common_size = 0
   Loop for i from 0 to SIZE
      If binary_search(second_array, first_array[i]) != -1
         common_array[common_size++] = first_array[i]
      End if
   End loop

   Return common_size
End
```

Begin find_two_largest(arr, size, first_largest, second_largest)
    *first_largest = INT_MIN
    *second_largest = INT_MIN

    Loop for i from 0 to SIZE
        If arr[i] > *first_largest
            *second_largest = *first_largest
            *first_largest = arr[i]
        Else if arr[i] > *second_largest
            *second_largest = arr[i]
        End if
    End loop
End

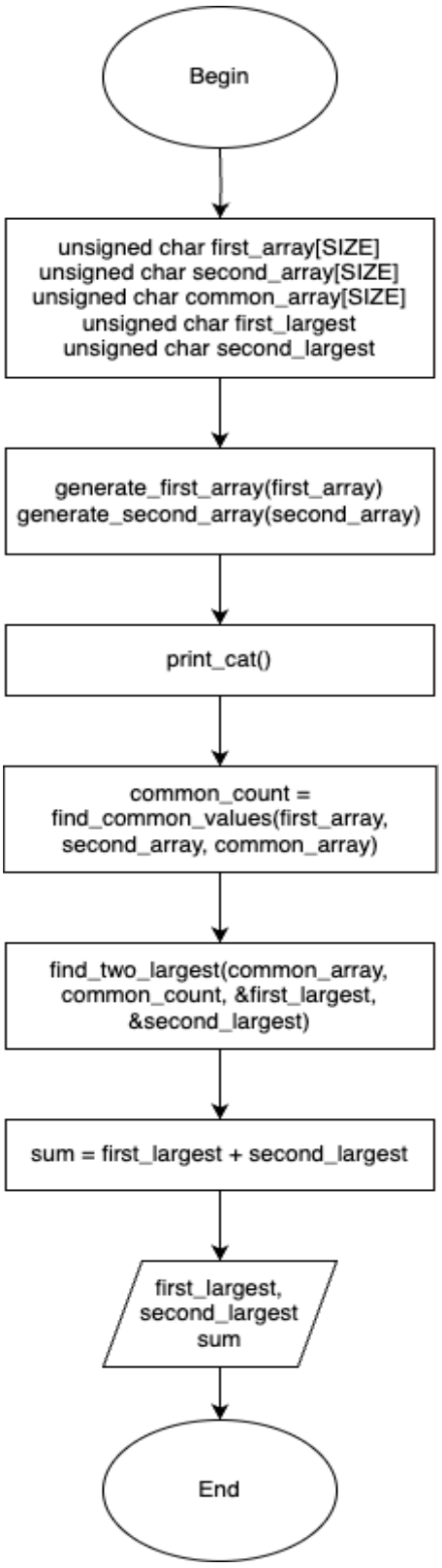
Begin generate_first_array(array)
    Loop for i from 0 to SIZE
        array[i] = 117 + i
    End loop
End

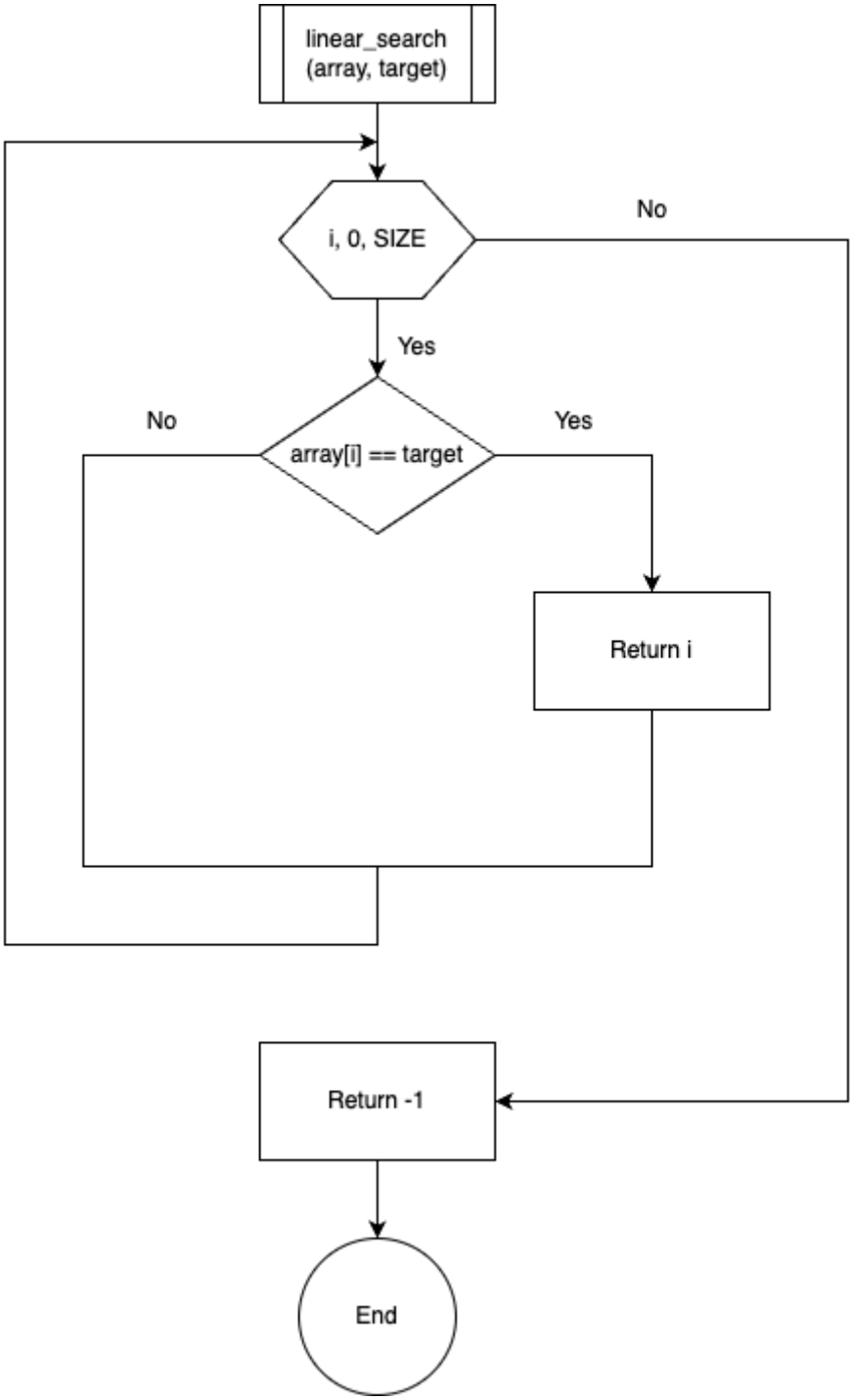
Begin generate_second_array(array)
    Loop for i from 0 to SIZE
        array[i] = 127 - 2 * i
    End loop
End

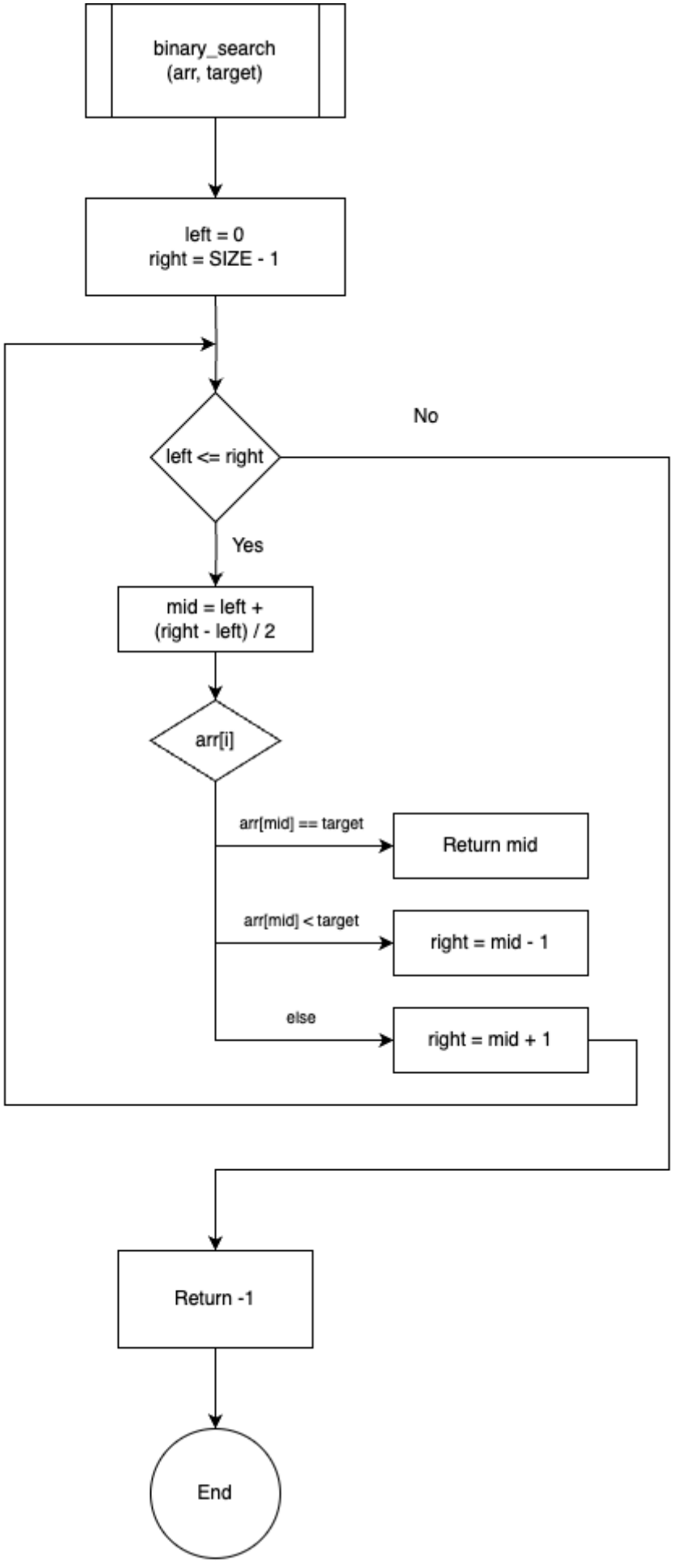
Begin print_cat

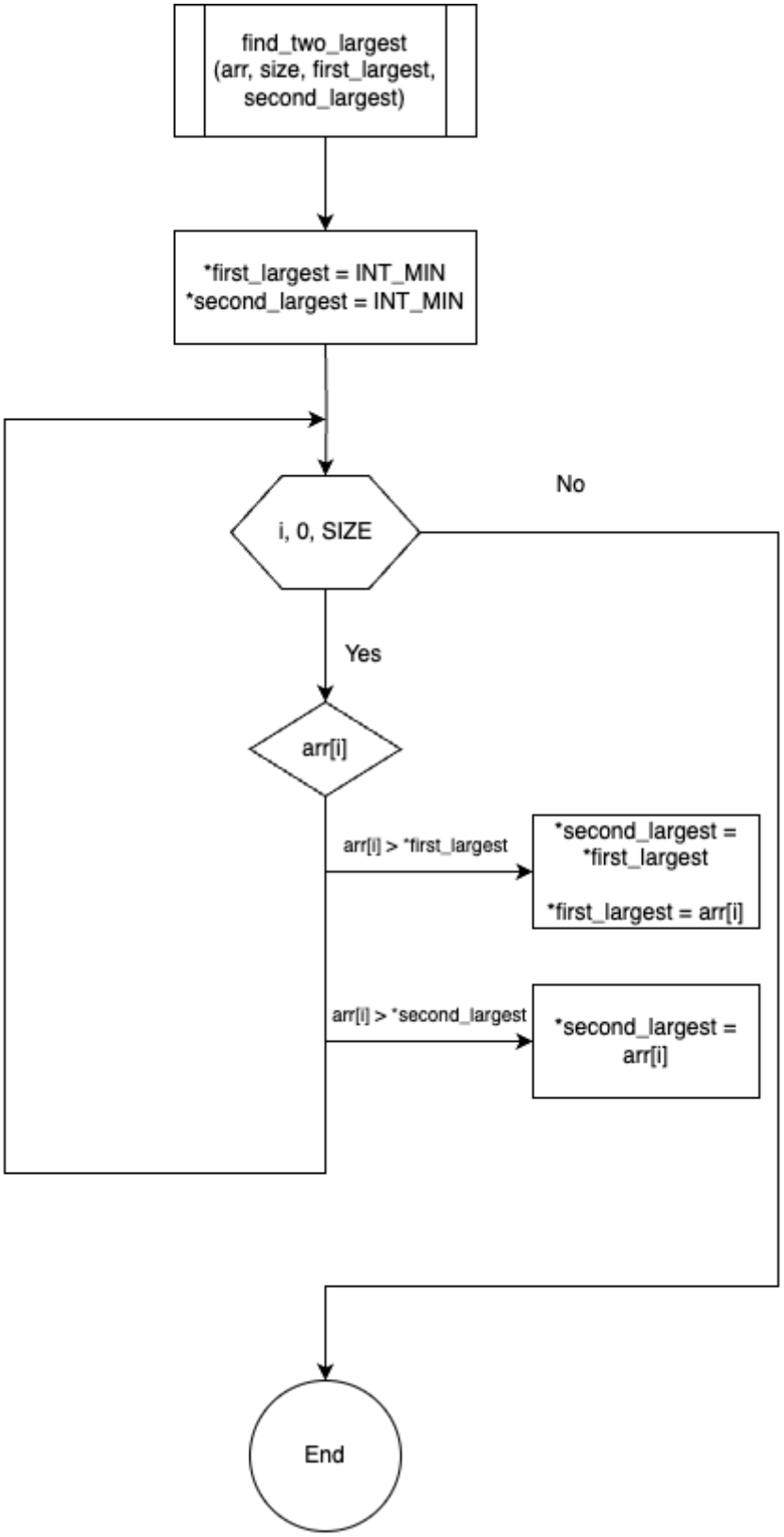
End

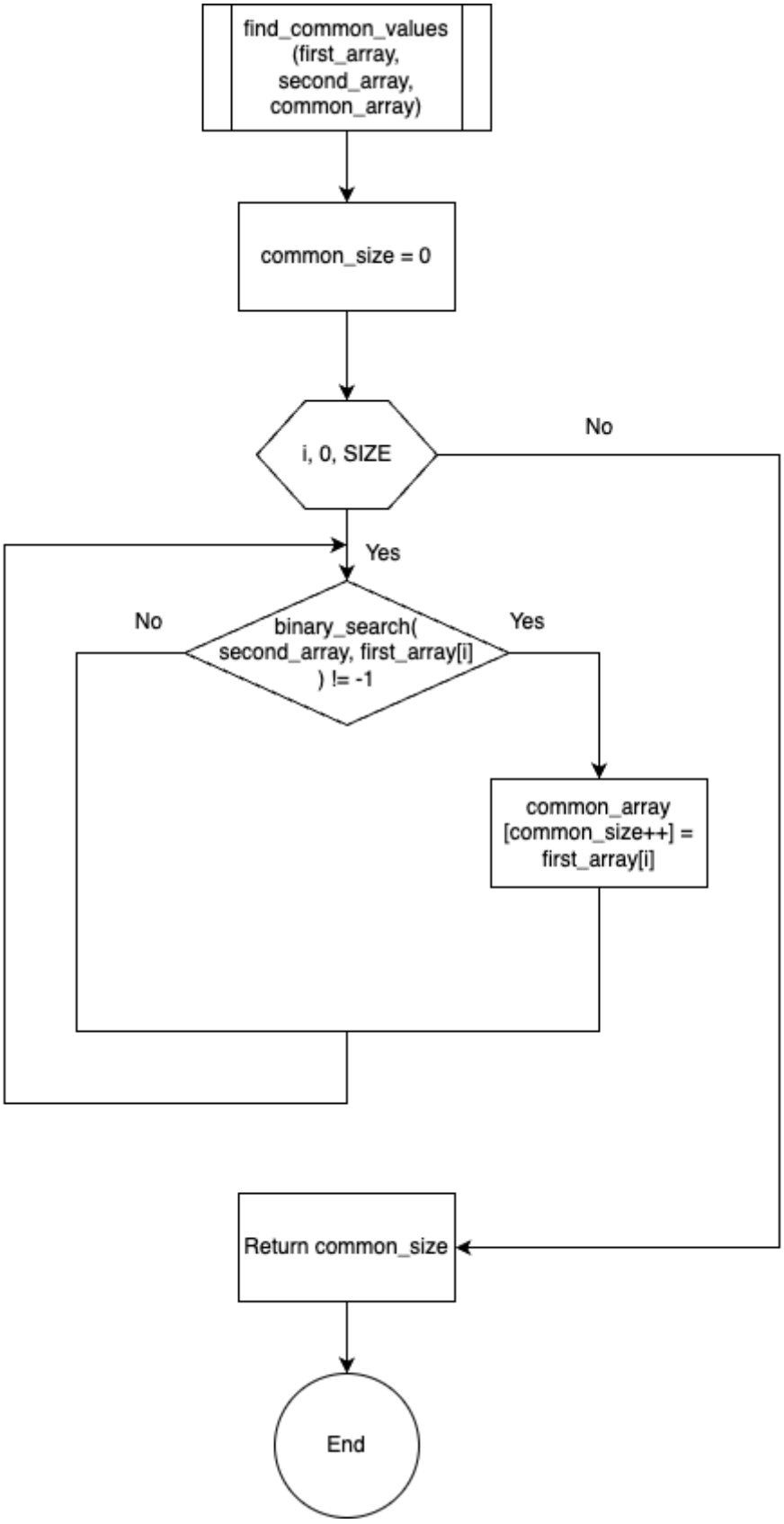
Блок-схема

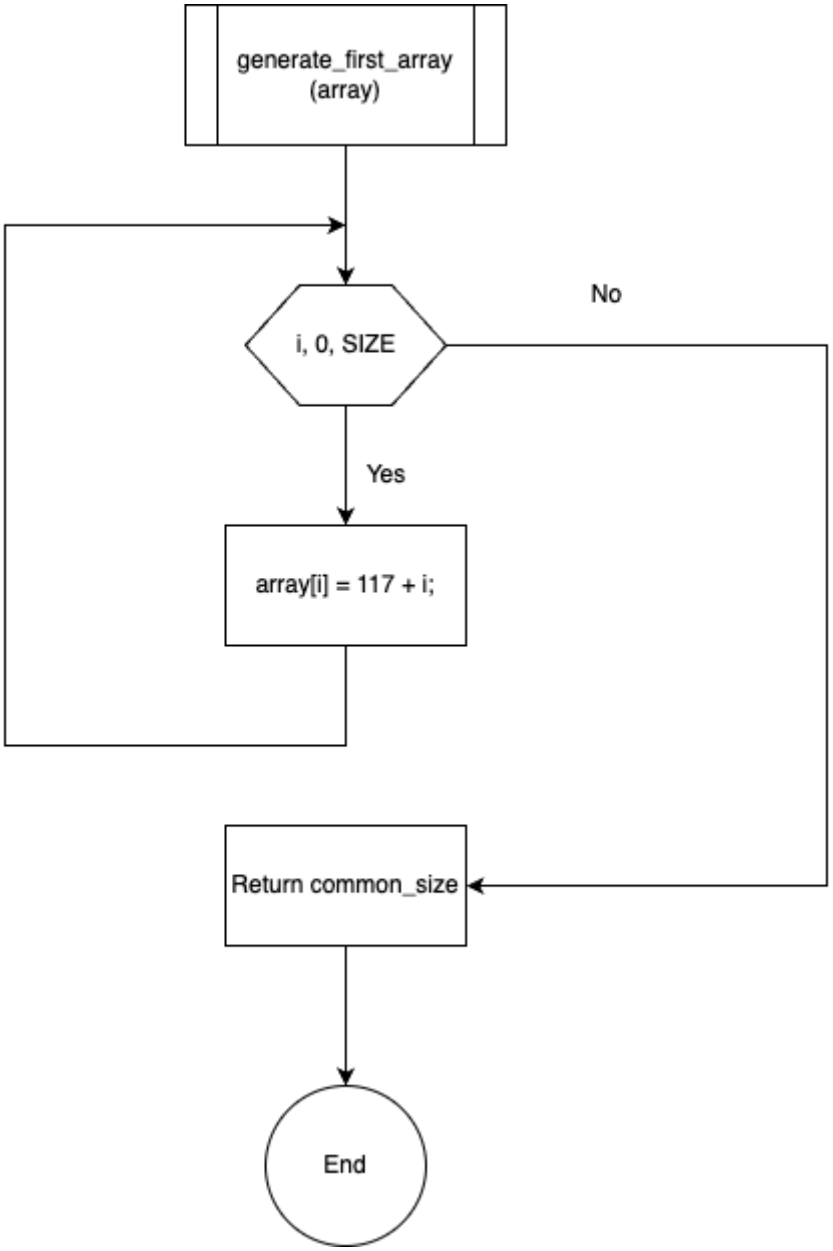

Begin
unsigned char first_array[SIZE]
unsigned char second_array[SIZE]
unsigned char common_array[SIZE]
unsigned char first_largest
unsigned char second_largest
generate_first_array(first_array)
generate_second_array(second_array)
print_cat()
common_count =
find_common_values(first_array,
second_array, common_array)
find_two_largest(common_array,
common_count, &first_largest,
&second_largest)
sum = first_largest + second_largest
first_largest,
second_largest
sum
End

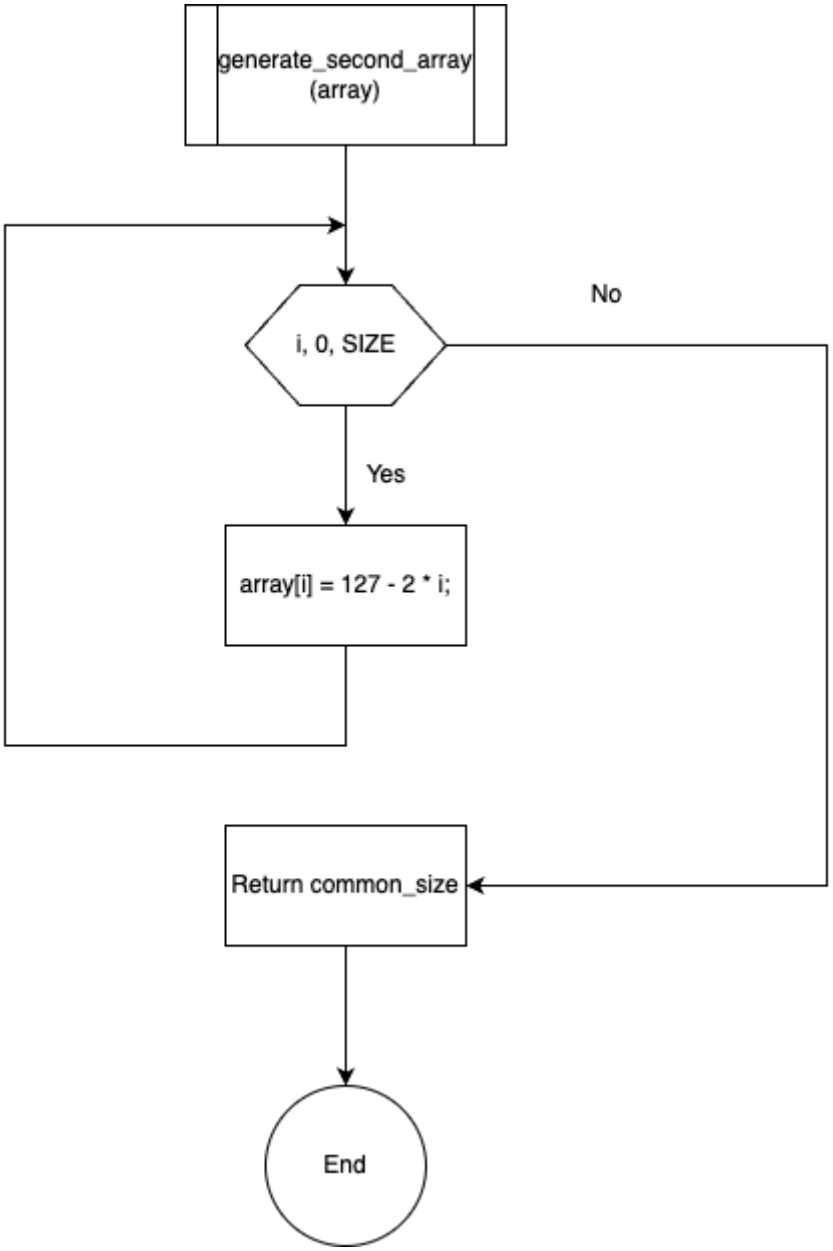

linear_search
(array, target)
i, 0, SIZE
No
Yes
No
array[i] == target
Yes
Return i
Return -1
End


binary_search
(arr, target)
left = 0
right = SIZE - 1
left <= right
No
Yes
mid = left +
(right - left) / 2
arr[i]
arr[mid] == target
Return mid
arr[mid] < target
right = mid - 1
else
right = mid + 1
Return -1
End


find_two_largest
(arr, size, first_largest,
second_largest)
*first_largest = INT_MIN
*second_largest = INT_MIN
i, 0, SIZE
No
Yes
arr[i]
arr[i] > *first_largest
*second_largest =
*first_largest
*first_largest = arr[i]
arr[i] > *second_largest
*second_largest =
arr[i]
End


find_common_values
(first_array,
second_array,
common_array)
common_size = 0
i, 0, SIZE
No
Yes
No
binary_search(
second_array, first_array[i]
) != -1
Yes
common_array
[common_size++] =
first_array[i]
Return common_size
End


generate_first_array
(array)
i, 0, SIZE
No
Yes
array[i] = 117 + i;
Return common_size
End


generate_second_array
(array)
i, 0, SIZE
No
Yes
array[i] = 127 - 2 * i;
Return common_size
End

Тестування алгоритму

First	117	118	119	120	121	122	123	124	125	126
Second	127	125	123	121	119	117	115	113	111	109
Common	117	119	121	123	125					

First largest	125
Second largest	123
Sum	248

Код мовою C

```c
#define SIZE 10

int linear_search(unsigned char* array, unsigned char target) {
    for (int i = 0; i < SIZE; i++) {
        if (array[i] == target) {
            return i;
        }
    }

    return -1;
}

int binary_search(unsigned char* arr, unsigned char target) {
    int left = 0;
    int right = SIZE - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
```

```
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }

    return -1;
}

int find_common_values(
    unsigned char* first_array,
    unsigned char* second_array,
    unsigned char* common_array
) {
    int common_size = 0;
    for (int i = 0; i < SIZE; i++) {
        if (binary_search(second_array, first_array[i]) != -1) {
            common_array[common_size++] = first_array[i];
        }
    }

    return common_size;
}

void find_two_largest(
    unsigned char* arr,
    int size,
    unsigned char* first_largest,
    unsigned char* second_largest
```

```
) {
    *first_largest = INT_MIN;
    *second_largest = INT_MIN;

    for (int i = 1; i < size; i++) {
        if (arr[i] > *first_largest) {
            *second_largest = *first_largest;
            *first_largest = arr[i];
        } else if (arr[i] > *second_largest) {
            *second_largest = arr[i];
        }
    }
}

void generate_first_array(unsigned char* array) {
    for (int i = 0; i < SIZE; i++) {
        array[i] = 117 + i;
    }
}

void generate_second_array(unsigned char* array) {
    for (int i = 0; i < SIZE; i++) {
        array[i] = 127 - 2 * i;
    }
}

int main() {
    unsigned char first_array[SIZE];
    unsigned char second_array[SIZE];
    unsigned char common_array[SIZE];
    unsigned char first_largest;
    unsigned char second_largest;
```

```
    generate_first_array(first_array);
    generate_second_array(second_array);

    print_cat();

    int common_count = find_common_values(first_array,
second_array, common_array);
    find_two_largest(common_array, common_count, &first_largest,
&second_largest);
    unsigned char sum = first_largest + second_largest;

    printf("First largest: %d -> %c\n", first_largest,
first_largest);
    printf("Second largest: %d -> %c\n", second_largest,
second_largest);
    printf("Sum of largest values: %d -> %c\n", sum, sum);

    return 0;
}
```

Висновок: Дослідив алгоритми пошуку в послідовностях та набув практичних навичок їх використання.