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 27

Виконав студент ІП-11 Савенко Олексій Андрійович
(шифр, прізвище, ім'я, по батькові)

Перевірив Мартинова О. П.
(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 27

Індивідуальне завдання

27	95 + i	105 - i	Суму елементів, коди яких більше 101

Постановка задачі

Результатом завдання є обчислення суми кодів елементів 3-го масиву, який побудований за допомогою спільних елементів з двох перших символьних масивів, які потрібно також створити та заповнити відповідними значеннями за допомогою таблиці ASCII.

Математична модель

main()

Змінна	Тип	Ім’я	Призначення
Змінна для визначення розміру масивів	Константа цілочисельного типу	**size**	Вхідні дані
Перший масив символьного типу	Масив символьного типу	**array_first**	Проміжні дані
Другий масив символьного типу	Масив символьного типу	**array_second**	Проміжні дані
Третій масив символьного типу	Масив символьного типу	**array_third**	Проміжні дані

Сума елементів коди яких більше за 101	Цілий	**sum**	Вихідні дані

Функції

Generation1,2()

Змінна	Тип	Ім’я	Призначення
Значення для присвоєння і-му елементу в масиві, відповідного значення	Символьний	***array**	Проміжні дані, формальний параметр
Змінна, яка вказує розмірність відповідного масиву	Константа цілочисельного типу	**size**	Проміжні дані, формальний параметр
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**i**	Проміжні дані

Output()

Змінна	Тип	Ім’я	Призначення
Змінна для виведення значення елементів масиву	Символьний	***array**	Проміжні дані, формальний параметр
Змінна, яка вказує розмірність відповідного масиву	Константа цілочисельного типу	**size**	Проміжні дані, формальний параметр
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**i**	Проміжні дані

Generation 3()

Змінна	Тип	Ім’я	Призначення
Змінна, яка приймає значення першого масива	Масив символьного типу	**arr[]**	Проміжні дані, формальний параметр
Змінна, яка приймає значення другого масива	Масив символьного типу	**arr2[]**	Проміжні дані, формальний параметр
Змінна, яка приймає значення	Масив символьного типу	**arr3[]**	Проміжні дані, формальний

третього масива			параметр
Змінна, яка вказує розмірність відповідного масиву	Константа цілочисельного типу	**size**	Проміжні дані, формальний параметр
Лічильник арифметичного циклу задля здійснення перебору елементів першого масиву	Цілий	**i**	Проміжні дані
Лічильник арифметичного циклу задля здійснення перебору елементів другого масиву	Цілий	**j**	Проміжні дані
Змінна, яка вказує розмірність третього масиву	Цілий	**arr3size**	Проміжні дані

Sum()

Змінна	Тип	Ім’я	Призначення
Змінна, яка приймає значення	Масив символьного типу	**array[]**	Проміжні дані, формальний

третього масива			параметр
Змінна, яка вказує розмірність відповідного масиву	Константа цілочисельного типу	**size**	Проміжні дані, формальний параметр
Лічильник арифметичного циклу задля здійснення операцій з масивом	Цілий	**i**	Проміжні дані
Змінна для отримання значення суми кодів символів елементів 3 го масиву	Цілий	**sum**	Вихідні дані

Пояснення логіки алгоритму:

Крок 1. Визначимо та охарактеризуємо основні дії

Крок 2. Деталізуємо дію оголошення трьох масивів та їх розмірності

Крок 3 Деталізуємо дію заповнення першого масиву

Крок 4. Деталізуємо дію заповнення другого масиву

Крок 5. Деталізуємо дію виведення елементів першого масиву

Крок 6. Деталізуємо дію виведення елементів другого масиву

Крок 7. Деталізуємо дію заповнення третього третього масиву

Крок 8. Деталізуємо дію обчислення суми кодів відповідних елементів

Псевдокод алгоритму:

Крок 1.

Початок

Деталізуємо дію оголошення трьох масивів та їх розмірності

Деталізуємо дію заповнення першого масиву

Деталізуємо дію заповнення другого масиву

Деталізуємо дію виведення елементів першого масиву

Деталізуємо дію виведення елементів другого масиву

Деталізуємо дію заповнення третього третього масиву

Деталізуємо дію обчислення суми кодів відповідних елементів

Виведення sum

Кінець

Крок 2.

Початок

const size = 10,

array_first[size],

array_second[size],

array_third[size]

Деталізуємо дію заповнення першого масиву

Деталізуємо дію заповнення другого масиву

Деталізуємо дію виведення елементів першого масиву

Деталізуємо дію виведення елементів другого масиву

Деталізуємо дію заповнення третього третього масиву

Деталізуємо дію обчислення суми кодів відповідних елементів

Виведення sum

Кінець

Крок 3.

Початок

const size = 10

array_first[size]

array_second[size]

array_third[size]

Generation1(array_first,size)

Деталізуємо дію заповнення другого масиву

Деталізуємо дію виведення елементів першого масиву

Деталізуємо дію виведення елементів другого масиву

Деталізуємо дію заповнення третього третього масиву

Деталізуємо дію обчислення суми кодів відповідних елементів

Виведення sum

Кінець

Generation1(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком** 1

*array = 95 + i

array++

все повторити

Кінець

Крок 4.

Початок

const size = 10

array_first[size]

array_second[size]

array_third[size]

Generation1(array_first,size)

Generation2(array_second,size)

Деталізуємо дію виведення елементів першого масиву

Деталізуємо дію виведення елементів другого масиву

Деталізуємо дію заповнення третього третього масиву

Деталізуємо дію обчислення суми кодів відповідних елементів

Виведення sum

Кінець

Generation1(*array,size**)**

повторити
для i **від** 1 **до** size **із кроком** 1

*array = 95 + i
array++

все повторити

Кінець

Generation2(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком** 1

*array = 105 - i

array++

все повторити

Кінець

Крок 4.

Початок

const size = 10

array_first[size]

array_second[size]

array_third[size]

Generation1(array_first,size)

Generation2(array_second,size)

Output(array_first,size)

Деталізуємо дію виведення елементів другого масиву

Деталізуємо дію заповнення третього третього масиву

Деталізуємо дію обчислення суми кодів відповідних елементів

Виведення sum

Кінець

Generation1(*array,size**)**

повторити
для i **від** 1 **до** size **із кроком** 1

*array = 95 + i
array++

все повторити

Кінець

Generation2(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком** 1

*array = 105 - i

array++

все повторити

Кінець

Output(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком 1**

Виведення *array

array++

все повторити

Кінець

Крок 6.

Початок

const size = 10

array_first[size]

array_second[size]

array_third[size]

Generation1(array_first,size)

Generation2(array_second,size)

Output(array_first,size)

Output(array_second,size)

Деталізуємо дію заповнення третього третього масиву

Деталізуємо дію обчислення суми кодів відповідних елементів

Виведення sum

Кінець

Generation1(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком** 1

*array = 95 + i

array++

все повторити

Кінець

Generation2(*array,size)

повторити

для i **від** 1 **до** size **із кроком** 1

*array = 105 - i

array++

все повторити

Кінець

Output(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком 1**

Виведення *array

array++

все повторити

Кінець

Output(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком 1**

Виведення *array

array++

все повторити

Кінець

Крок 7.

Початок

const size = 10

array_first[size]

array_second[size]

array_third[size]

Generation1(array_first,size)

Generation2(array_second,size)

Output(array_first,size)

Output(array_second,size)

Generation3(array_first, array_second, array_third, size)

Деталізуємо дію обчислення суми кодів відповідних елементів

Виведення sum

Кінець

Generation1(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком** 1

*array = 95 + i

array++

все повторити

Кінець

Generation2(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком** 1

*array = 105 - i

array++

все повторити

Кінець

Output(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком 1**

Виведення *array

array++

все повторити

Кінець

Output(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком 1**

Виведення *array

array++

все повторити

Кінець

Generation3(arr[], arr2[], arr3[], size**)**

arr3size = 0

повторити

для i **від** 1 **до** size **із кроком** 1

повторити

для j **від** 1 **до** size **із кроком** 1

Якщо arr[i] == arr2[j]

то

arr3[arr3size] = arr[i]

arr3size++

все повторити

інакше

все повторити

все повторити

Кінець

Крок 8.

Початок

const size = 10

array_first[size]

array_second[size]

array_third[size]

Generation1(array_first,size)

Generation2(array_second,size)

Output(array_first,size)

Output(array_second,size)

Generation3(array_first, array_second, array_third, size)

sum = Sum(array_third,size)

Виведення sum

Кінець

Generation1(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком** 1

*array = 95 + i

array++

все повторити

Кінець

Generation2(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком** 1

*array = 105 - i

array++

все повторити

Кінець

Output(*array,size**)**

повторити

для i **від** 1 **до** size **із кроком 1**

Виведення *array

array++

все повторити

Кінець

Output(*array,size**)**

повторити
для i **від** 1 **до** size **із кроком 1**

Виведення *array
array++

все повторити

Кінець

Generation3(arr[], arr2[], arr3[], size**)**

arr3size = 0

повторити
для i **від** 1 **до** size **із кроком** 1

повторити
для j **від** 1 **до** size **із кроком** 1

Якщо arr[i] == arr2[j]

то

arr3[arr3size] = arr[i]

arr3size++

все повторити

інакше

все повторити

все повторити

Кінець

Sum(array[], size**)**

sum = 0

повторити

для i **від** 1 **до** size **із кроком** 1

Якщо array[i] > 101

то

sum += array[i]

все повторити

інакше

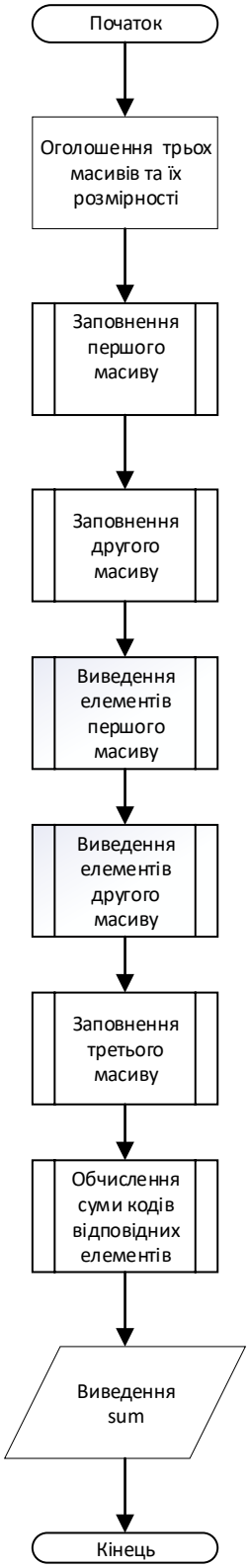
все повторити

Повернути sum

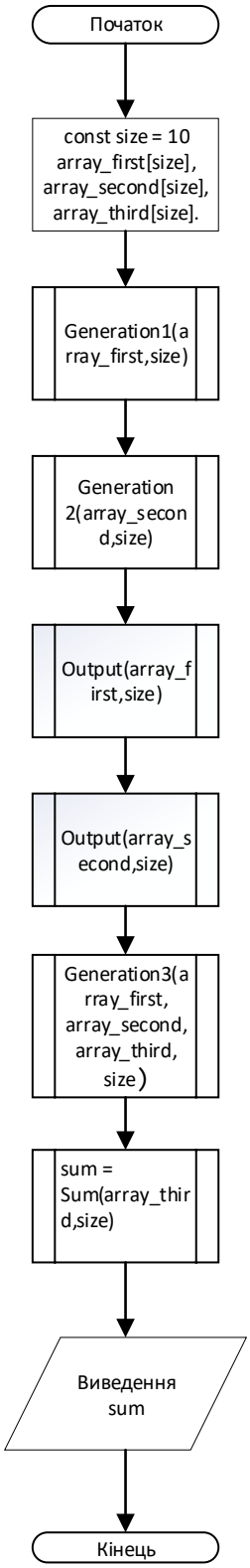
Кінець

Блок-схеми алгоритму

Блок-схема опису дій основної програми:

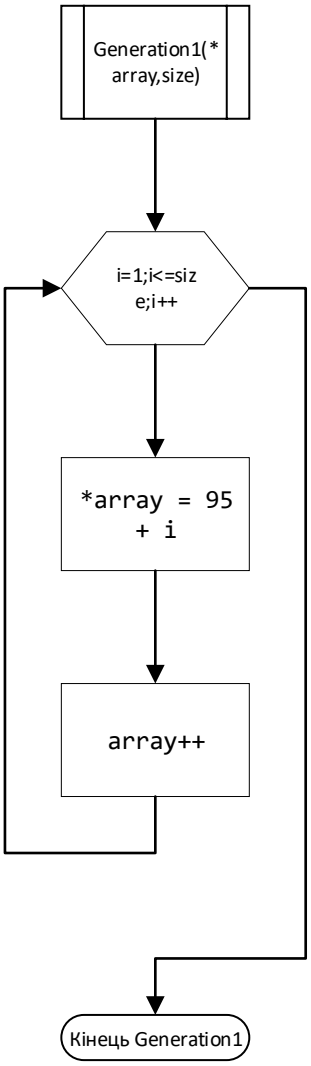


Блок-схема деталізації дій основної програми:

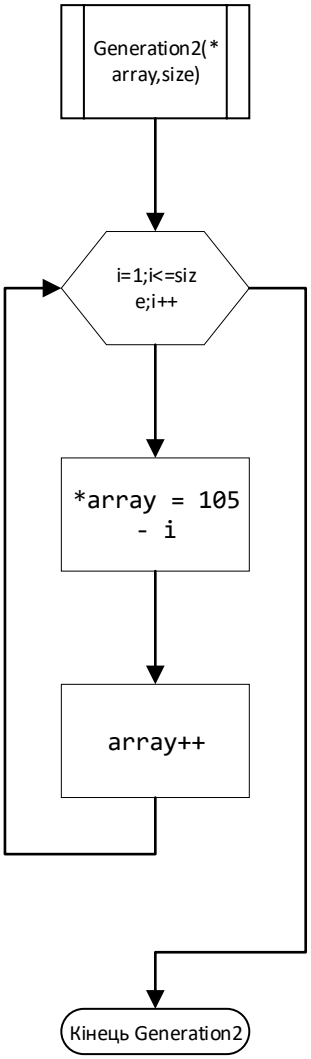


Блок-схеми функцій:

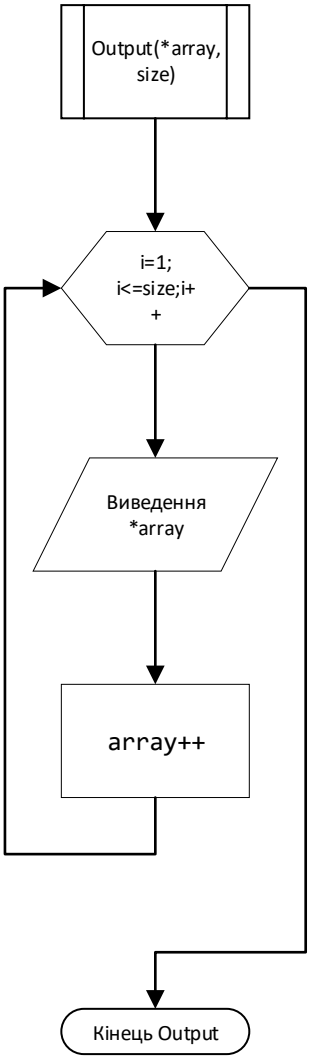
Generation1():



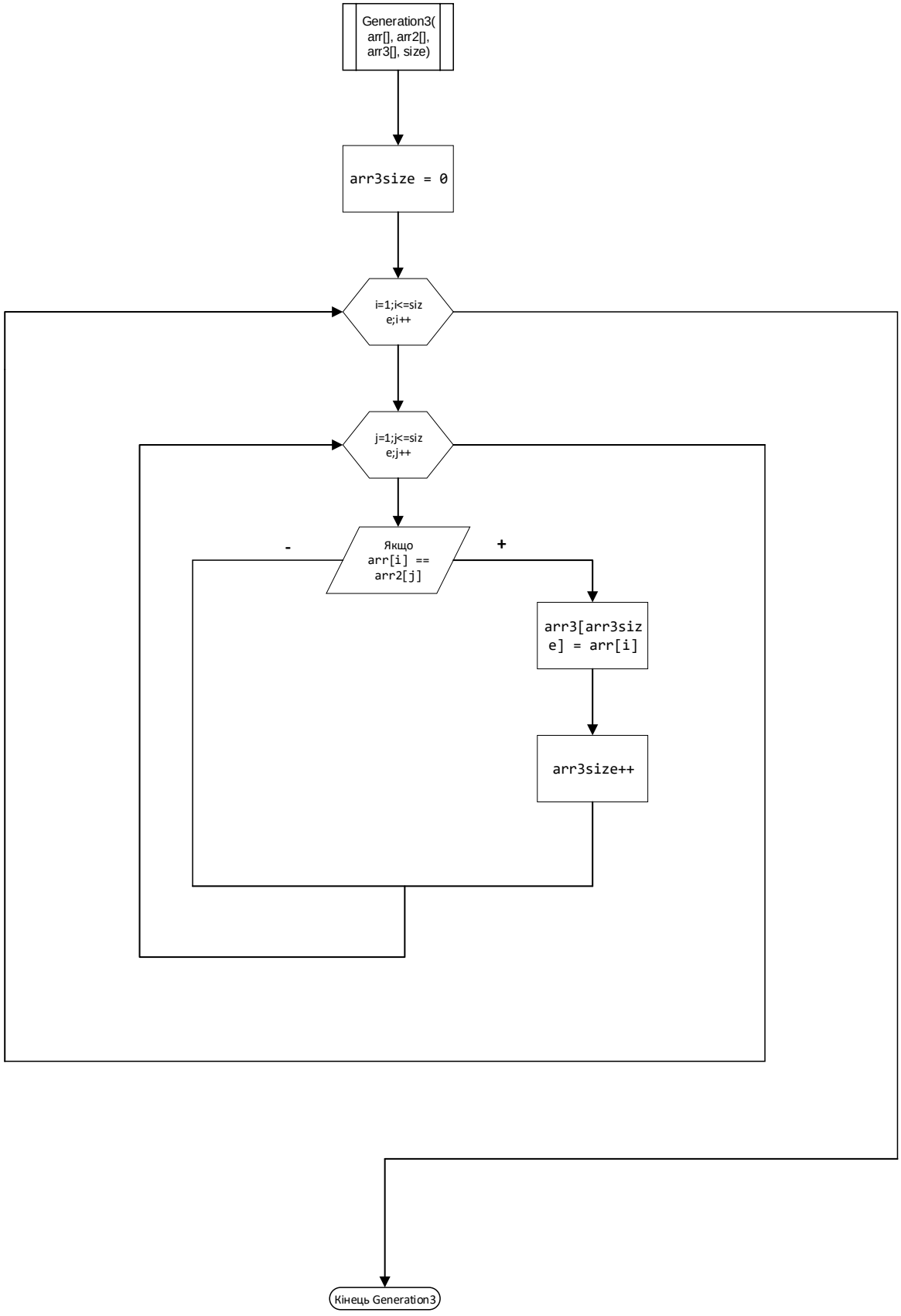
Generation2():



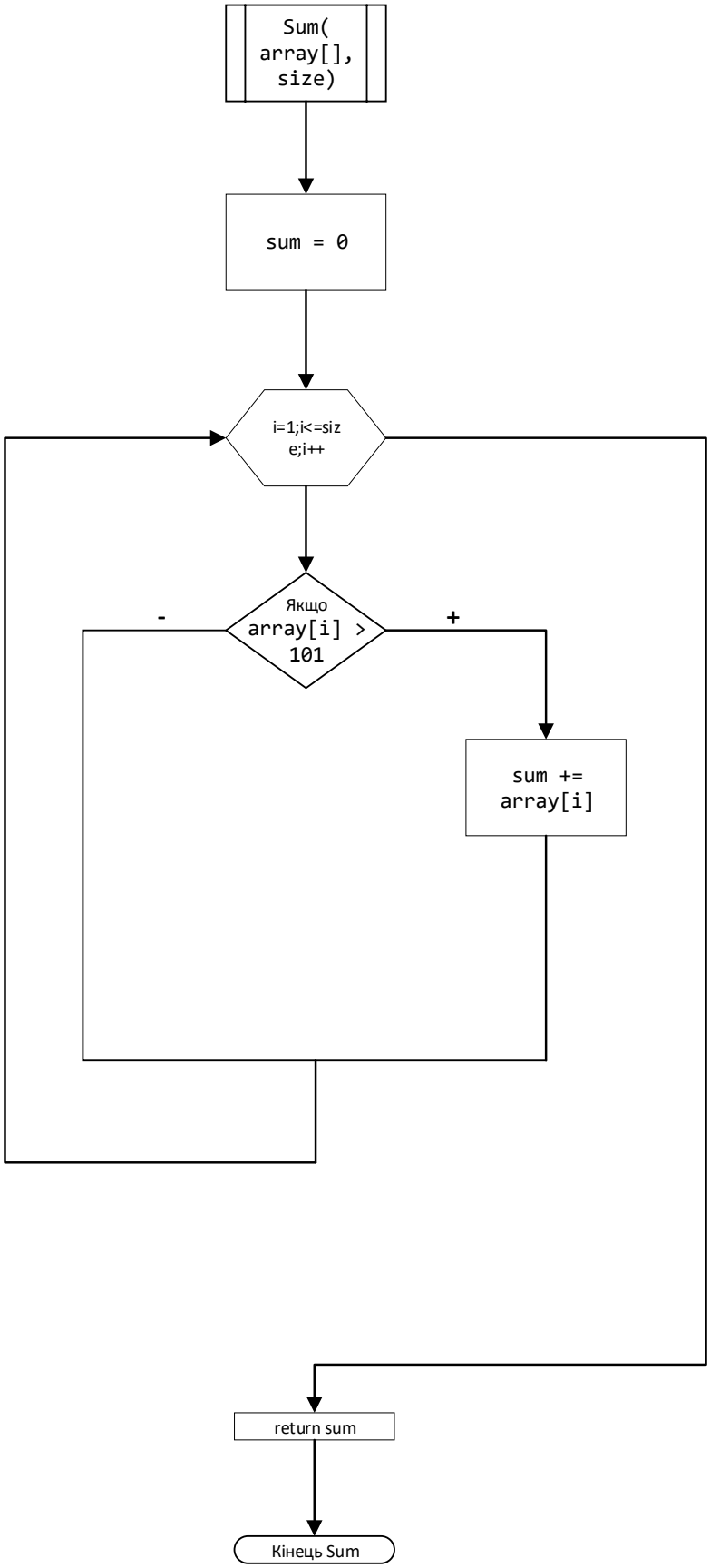
Output():


Output(*array, size)
i=1; i<=size;i++
Виведення *array
array++
Кінець Output

Generation3():



Sum():

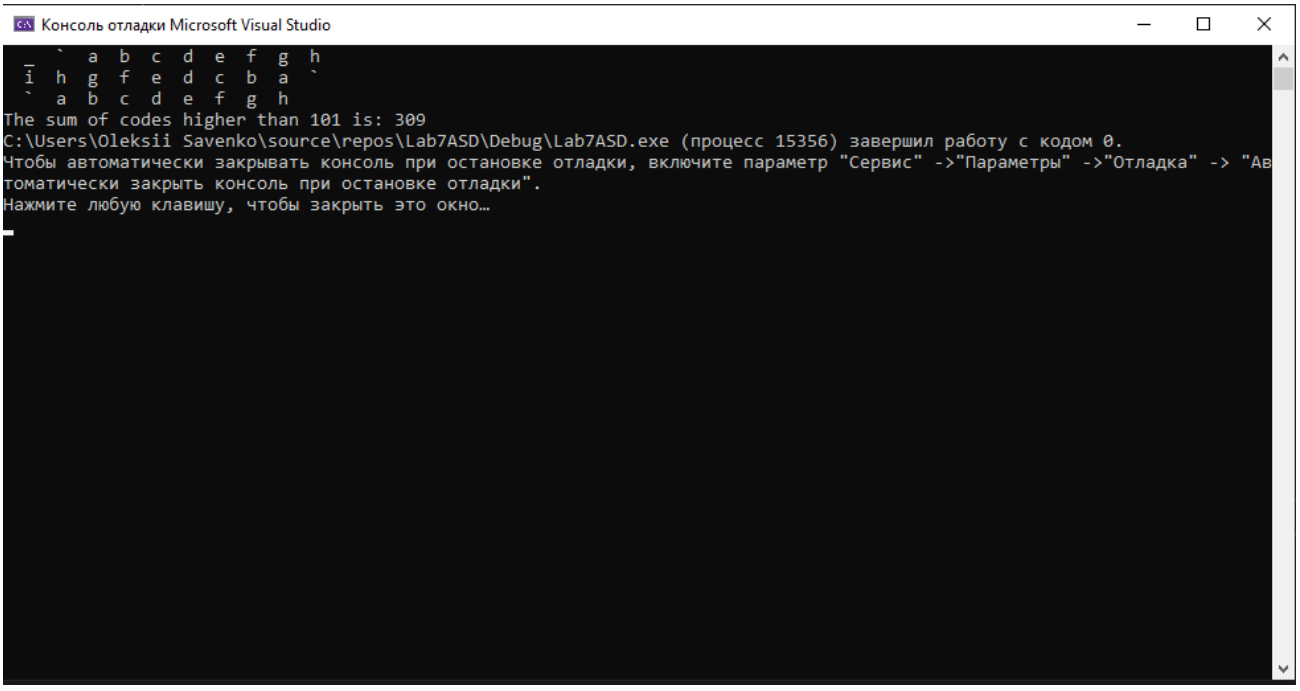

Sum(array[], size)
sum = 0
i=1;i<=siz e;i++
Якщо array[i] > 101
-
+
sum += array[i]
return sum
Кінець Sum

Код програми на мові C++:

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


void Generation1(char*, int);
void Generation2(char*, int);
void Output(char*, int);
void Generation3(char*, char*, char*, int);
int  Sum(char*, int);
int main()
{
    const int size = 10; int sum;
    char array_first[size], array_second[size], array_third[size] = { '\0' };
    Generation1(array_first, size);
    Generation2(array_second, size);
    Output(array_first, size); cout << endl;
    Output(array_second, size); cout << endl;

    Generation3(array_first, array_second, array_third, size);
    Output(array_third, size);

    sum = Sum(array_third, size); cout << endl << "The sum of codes higher than 101 is: " <<
sum;

}




void Generation1(char* array, int size) {

    for (int i = 0; i < size; i++) {
        *array = 95 + i;
        array++;
    }
}

void Generation2(char* array, int size) {

    for (int i = 0; i < size; i++) {
        *array = 105 - i;
        array++;
    }
}

void Output(char* array, int size) {
    for (int i = 0; i < size; i++) {
        cout << setw(3) << *array;
        array++;

    }
}



void Generation3(char arr[], char arr2[], char arr3[], int size) {
```

```
    int arr3size = 0;
    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++) {
            if (arr[i] == arr2[j]) {
                arr3[arr3size] = arr[i];
                arr3size++;
            }
        }
    }
}


int  Sum(char array[], int size) {
        int sum = 0;
    for (int i = 0; i < size; i++) {
        if (array[i] > 101)
            sum += array[i];
    }
    return  sum;
}
```

Випробування алгоритму на мові програмування C++:



Випробування алгоритму:

Блок	Дія
	Початок
1. Оголошення трьох масивів та їх розмірності	const size = 10, array_first[size] array_second[size], array_third[size]
2. Заповнення першого масиву **Generation1()**	array_first[size]={ _ , `,a, b, c, d, e, f, g, h}
3. Заповнення другого масиву **Generation2()**	array_second[size]={ i, h, g, f, e, d, c, b, a, `}
4. Виведення елементів першого масиву **Output()**	**Виведення** array_first[size]**:**{ _ , `,a, b, c, d, e, f, g, h}
5. Виведення елементів другого масиву **Output()**	**Виведення** array_second[size]**: {**i, h, g, f, e, d, c, b, a, `}
6. Заповнення третього масиву **Generation3()**	array_third[size]={ `, a, b, c, d, e, f, g, h}
7. Обчислення суми кодів відповідних елементів **Sum()**	sum=309
8.	**Виведення: "**The sum of codes higher than 101 is: 309**"**
	Кінець

Висновок

Я дослідив методи послідовного пошуку у впорядкованих і невпорядкованих послідовностях та набув практичних навичок їх використання під час складання програмних специфікацій. Була розроблена постановка задачі де була пояснена логіка алгоритму, математична модель основної програми та підпрограм, псевдокод де розписаний алгоритм, а також блок-схеми основної програми та підпрограм які застосовуються при роботі алгоритму. Для розв’язання поставленого завдання було застосовано лінійний пошук для послідовного порівняння кожного елементу першої матриці з кожним елементом другої, була створена третя матриця яка була заповнена елементами спільними елементами двох перших, у підсумку чого була підрахована сума відповідних кодів. Для перевірки вірності алгоритму було проведено його випробування, матриці було заповнено символами кодів 95+i - 1 та другу 105-i, за допомогою операції лінійного пошуку було знайдено 9 елементів та сформовано 3 масив на їх основі. У підсумку було підраховано суму кодів елементів 3 масиву, коди яких більше 101, вона дорівнює 309, що є вірним. Отже, мій алгоритм є вірним і його можна використовувати для розв’язання завдань даного типу.