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 15

Виконав студент ІП-15, Костін Вадим Анатолійович
(шифр, прізвище, ім'я, по батькові)

Перевірив 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 15

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 (табл. 1).

3. Ініціювання третьої змінної рівними значеннями двох попередніх змінних.

4. Обробки третьої змінної згідно з варіантом.

№	Вираз для обчислення елемента		Знайти
	1-го масиву	2-го масиву	
15	43 - i	37 + i	Добуток елементів, коди яких більше 40

Постановка задачі

За умовою задачі дано два масиви. Третій масив буде мати в собі спільні елементи першого та другого масивів. Після створення третього масиву, робимо вказані в умові дії.

Математична модель

Генерувати перші два масиви A та B будемо за допомогою підпрограми get_AB(). Третій масив C згенеруємо за допомогою підпрограми get_C(). У ньому буде перевірятись рівність елементів масивів A та B, якщо елементи рівні, то один з них додаємо у масив C. За допомогою підпрограми Mult_C() знайдемо добуток елементів, коди яких більше за 40.

Змінна	Тип	Ім’я	Призначення
Перший масив	Символьний	A	Початкові дані
Другий масив	Символьний	B	Початкові дані

Третій масив	Символьний	C	Результат
Підпрограма що генерує масиви A та B	Процедура	get_AB()	Проміжні дані
Підпрограма що генерує масив C	Процедура	get_C()	Проміжні дані
Підпрограма що рахує добуток елементів масиву C, код яких більший за 40	Дійсний	Mult_C()	Результат
Лічильник масиву A	Цілочисельний	i	Проміжні дані
Лічильник масиву B	Цілочисельний	j	Проміжні дані
Кількість елементів у масиві C	Цілочисельний	l	Проміжні дані

*Для виразу x = x * y будемо використовувати x *= y*

Для виразу x = x + 1 будемо використовувати x++

Для перевірки на рівність будемо використовувати логічні вирази ==, !=,>,<

Крок 1 Деталізуємо основні дії

Крок 2 Деталізуємо дію генерації масивів A,B

Крок 3 Деталізуємо дію генерації масиву C

Крок 4 Деталізуємо дію знаходження добутку елементів масиву C, в яких код більший за 40

Псевдокод

Крок 1 **Початок** Ініціалізування A,B,C Деталізуємо дію генерації масивів A,B Деталізуємо дію генерації масиву C Деталізуємо дію знаходження добутку елементів масиву C, в яких код більший за 40 Виведення результатів **Кінець**	*Підпрограма get_AB()* *Підпрограма get_C()* *Підпрограма Mult_C()*
Крок 2 **Початок** Ініціалізування A,B,C get_AB(A,B) Деталізуємо дію генерації масиву C Деталізуємо дію знаходження добутку елементів масиву C, в яких код більший за 40 Виведення результатів **Кінець**	*Підпрограма get_AB(A,B)* **Початок** **Повторити** для i = 0, i < 10, i++ A[i] = 43 – i B[i] = 37 + i **Все повторити** **Кінець** *Підпрограма get_C()* *Підпрограма Mult_C()*
Крок 3 **Початок** Ініціалізування A,B,C get_AB(A,B) *get_C(C)* Деталізуємо дію знаходження добутку елементів масиву C, в яких код більший за 40 Виведення результатів **Кінець**	*Підпрограма get_AB(A,B)* **Початок** **Повторити** для i = 0, i < 10, i++ A[i] = 43 – i B[i] = 37 + i **Все повторити** **Кінець** *Підпрограма get_C(C)* **Початок** l = 0 **Повторити** для i = 0, i < 10, i++ **Повторити** для j = 0, i < 10, i++ **Якщо** A[i]==B[j] C[i] = A[i] l++ **Все якщо** **Все повторити** **Все повторити** **Кінець** *Підпрограма Mult_C()*

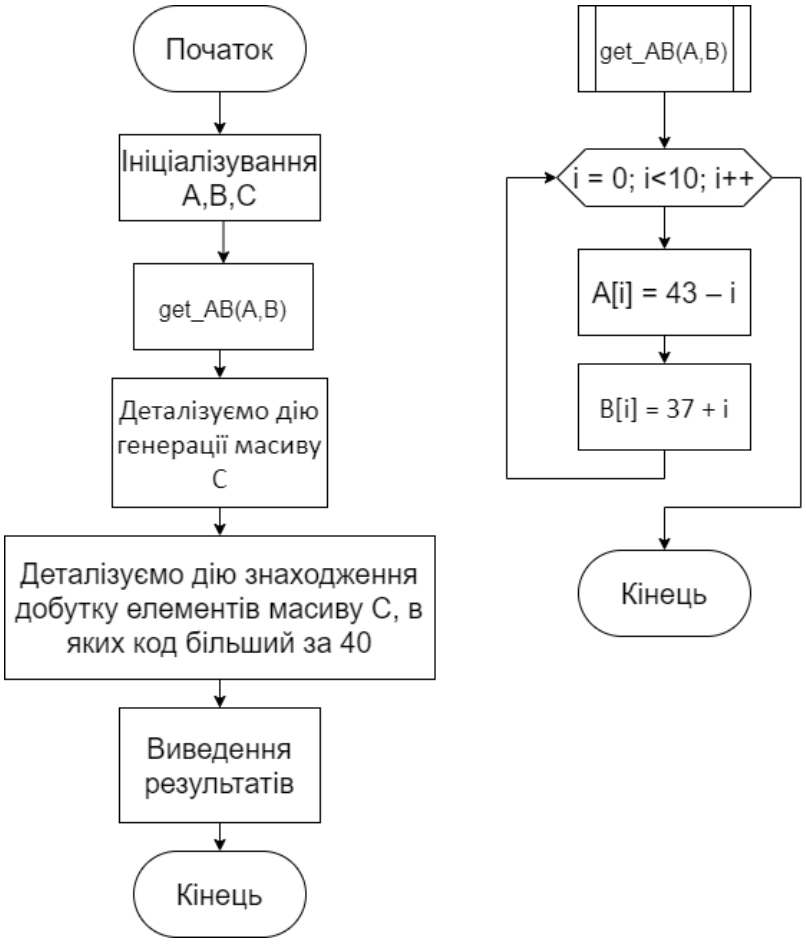
Крок 4
Початок
  Ініціалізування A,B,C
  get_AB(A,B)
  get_C(C)
  S = Mult_C(C)
  Виведення результатів
Кінець

Підпрограма get_AB(A,B)
Початок
  Повторити для i = 0, i < 10, i++
    A[i] = 43 – i
    B[i] = 37 + i
  Все повторити
Кінець
Підпрограма get_C(C,A,B)
Початок
  l = 0
  Повторити для i = 0, i < 10, i++
    Повторити для j = 0, i < 10, i++
      Якщо A[i]==B[j]
        C[i] = A[i]
        l++
      Все якщо
    Все повторити
  Все повторити
Кінець
Підпрограма Mult_C(C)
Початок
  s = 1
  Повторити для i = 0, i < 10, i++
    Якщо C[i] > 40
      s *= C[i]
    Все якщо
  Все повторити
  Повернути s
Кінець

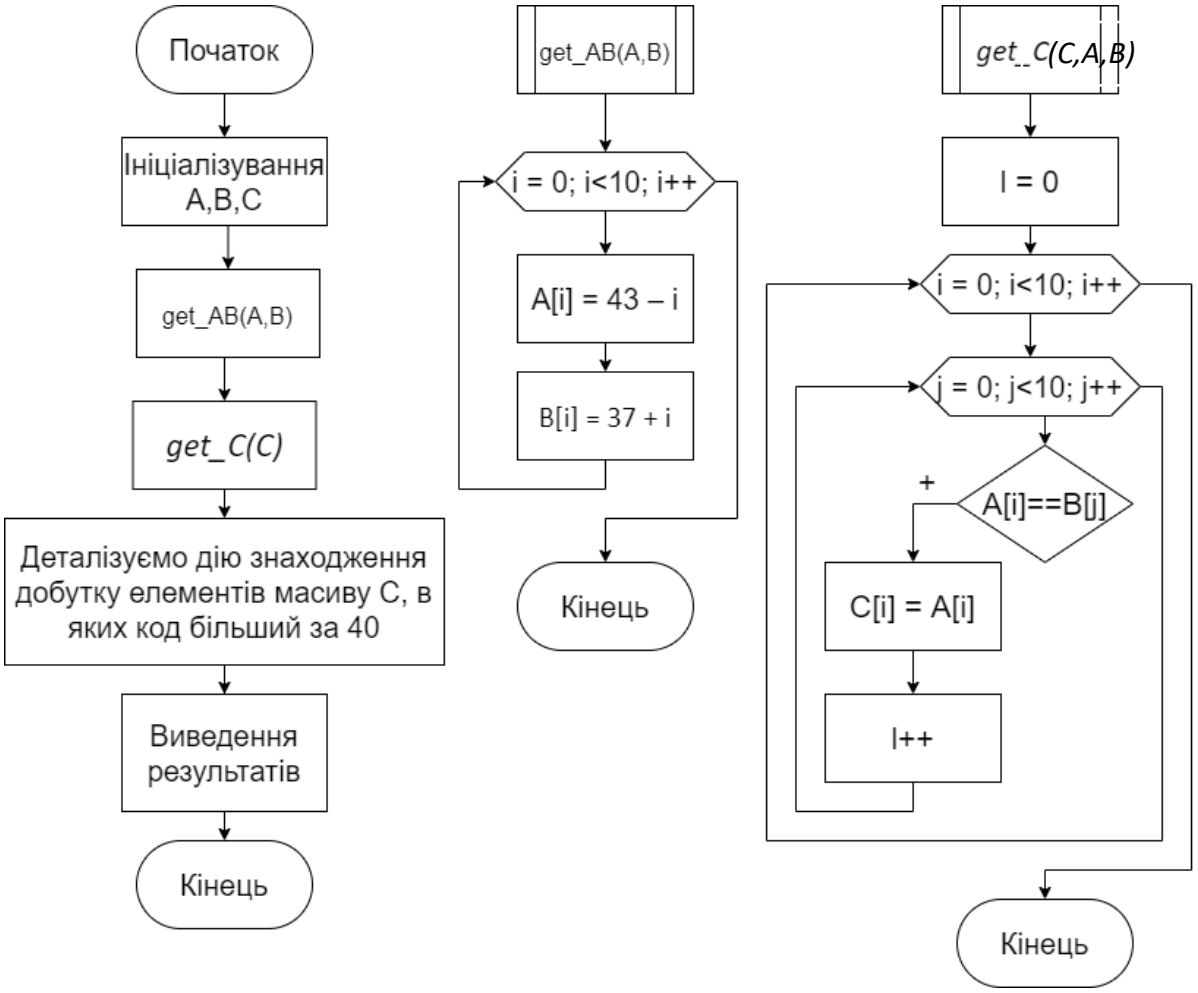
Блок-схеми

Крок 1

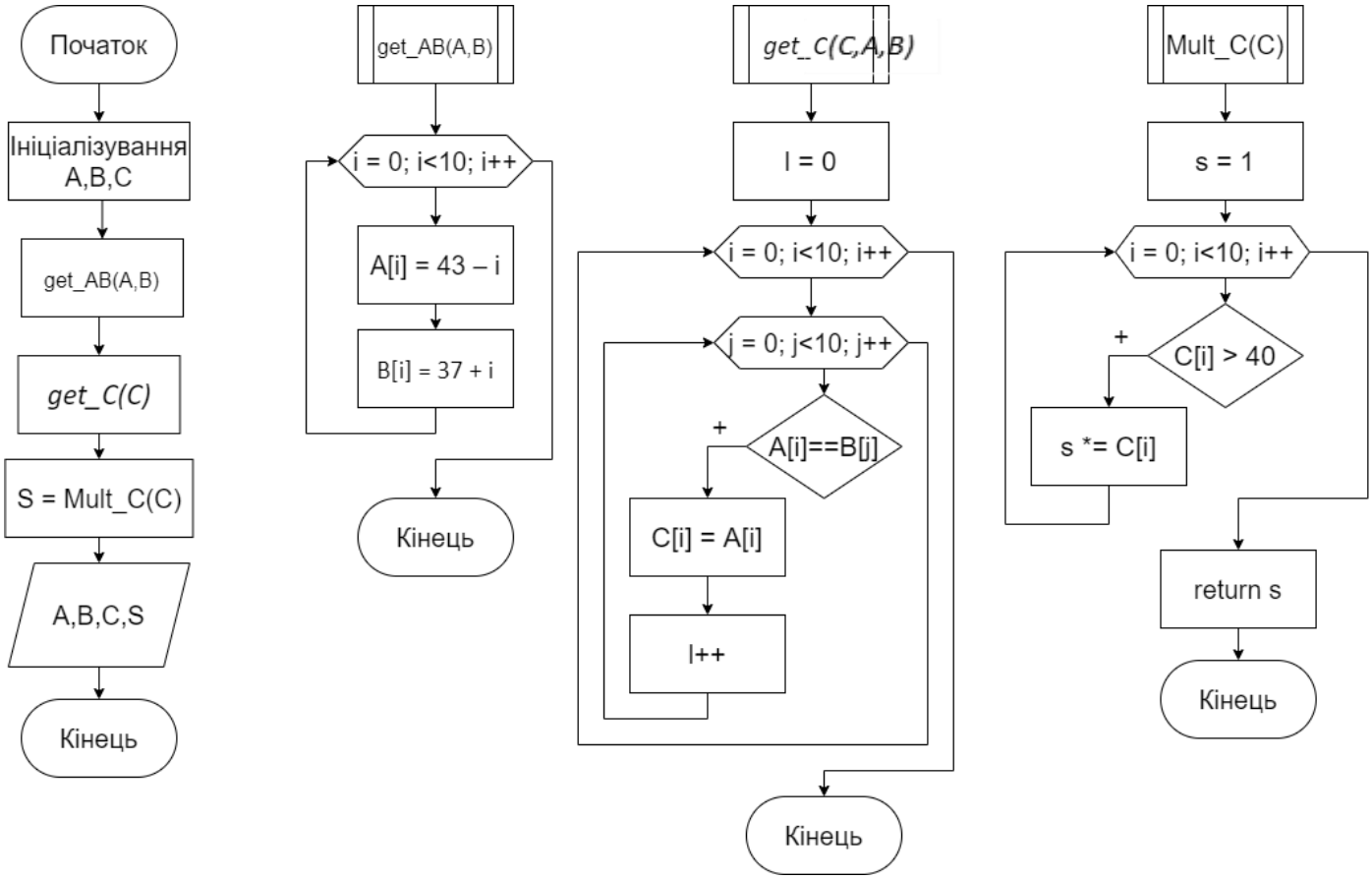
Крок 2



Крок 3


Початок
Ініціалізування A,B,C
get_AB(A,B)
get_C(C)
Деталізуємо дію знаходження добутку елементів масиву C, в яких код більший за 40
Виведення результатів
Кінець
get_AB(A,B)
i = 0; i<10; i++
A[i] = 43 – i
B[i] = 37 + i
Кінець
get_C(C,A,B)
I = 0
i = 0; i<10; i++
j = 0; j<10; j++
+
A[i]==B[j]
C[i] = A[i]
I++
Кінець

Крок 4

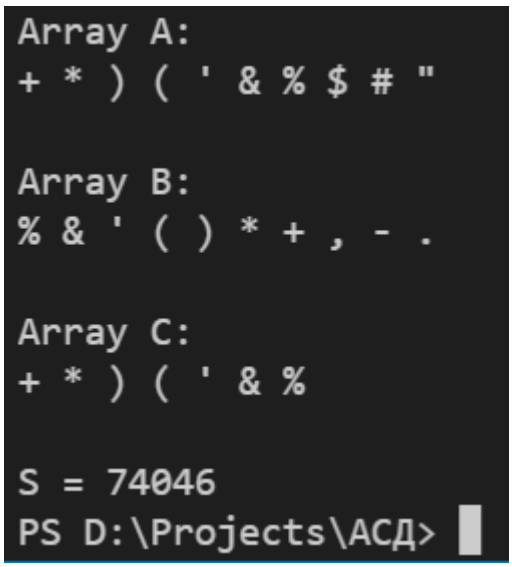

Початок
Ініціалізування A,B,C
get_AB(A,B)
get_C(C)
S = Mult_C(C)
A,B,C,S
Кінець
get_AB(A,B)
i = 0; i<10; i++
A[i] = 43 – i
B[i] = 37 + i
Кінець
get_C(C,A,B)
I = 0
i = 0; i<10; i++
j = 0; j<10; j++
+
A[i]==B[j]
C[i] = A[i]
I++
Кінець
Mult_C(C)
s = 1
i = 0; i<10; i++
+
C[i] > 40
s *= C[i]
return s
Кінець

Код програми

Lab7.cpp > main()

```cpp
#include <iostream>

using namespace std;

int l;

void get_AB(char A[], char B[]){
    for (int i = 0; i < 10; i++)
    {
        A[i] = 43 - i;
        B[i] = 37 + i;
    }
}

void get_C(char C[], char A[], char B[]){
    l = 0;
    for (int i = 0; i < 10; i++)
    {
        for (int j = 0; j < 10; j++)
        {
            if (A[i]==B[j]){
                C[l] = A[i];
                l++;
            }
        }
    }
}

int Mult_C(char C[]){
    int s = 1;
    for (int i = 0; i < 10; i++)
    {
        if (C[i]>40)
            s *= C[i];
    }
    return s;
}

int main(){
    char A[10], B[10], C[10] = {0,0,0,0,0,0,0,0,0,0};

    get_AB(A,B);
    get_C(C,A,B);
    int S = Mult_C(C);

    cout << "Array A:"<< endl;
    for (int i = 0; i < 10; i++)
        cout << A[i] << " ";

    cout << endl << endl << "Array B:"<< endl;
    for (int i = 0; i < 10; i++)
        cout << B[i] << " ";

    cout << endl << endl << "Array C:"<< endl;
    for (int i = 0; i <= l; i++)
        cout << C[i] << " ";

    cout << endl << endl << "S = " << S;
}
```

Результат роботи програми



Висновки

Протягом сьомої лабораторної роботи ми дослідили методи послідовного пошуку у впорядкованих і невпорядкованих послідовностях та набули практичних навичок їх використання під час складання програмних специфікацій.