# 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 лінійних алгоритмів» Варіант 2

Виконав студент Боровков Іван Ігорович

(шифр, прізвище, ім'я, по батькові)

Перевірив

( 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.

**Умова задачі** –

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.

**Постановка задачі**

За допомогою алгоритму лінійного пошуку визначити однакові елементи першого та другого масиву, сформувати з цих елементів третій масив та знайти добуток тих його елементів, коди яких менші за 40.

**Математична модель.**

**Складемо таблицю імен змінних:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Перший масив | Масив символів | first\_array | Вихідні дані |
| Другий масив | Масив символів | second\_array | Вихідні дані |
| Результуючий (третій) масив | Масив символів | third\_array(result\_array) | Вихідні дані |
| Розмір першого та другого масивів | Цілий | size | Допоміжна змінна |
| Розмір результуючого масиву | Цілий | third\_array\_size | Допоміжна змінна |
| Добуток значень результуючого масиву | Символ | result | Вихідні дані |

Для ініціалізації першої та другої послідовності використаємо арифметичний цикл, значення лічильника підставимо у задані в умові формули, і, таким чином, обчислимо коди символів, що є елементами цих послідовностей.

Для ініціалізації третьої послідовності використаємо два арифметичні цикли, що будуть попарно порівнювати символи з першої та другої послідовності та сформуємо з тих, що містяться у обох окремий масив, який і буде шуканим.

Для обчислення добутку елементів третього массиву, коди яких менші за 40 використаємо арифметичний цикл та альтернативну форму алгоритму розгалуження.

***Розв’язання:***

**Програмні специфікації** пишемо у псевдокоді та графічній формі у вигляді блок схеми та підпрограмм.

*Крок 1:* Визначимо основні дії ;

*Крок 2:* Ініціалізуємо змінні;

*Крок 3:* Ініціалізуємо першу послідовність;

*Крок 4:* Ініціалізуємо другу послідовність;

*Крок 5:* Ініціалізуємо третю послідовність;

*Крок 6:* Обчислимо добуток елементів третьої послідовності, коди яких менші за 40;

*Псевдокод*

*Крок 1*

**Початок**

Ініціалізуємо змінні;

Ініціалізуємо першу послідовність;

Ініціалізуємо другу послідовність;

Ініціалізуємо третю послідовність;

Обчислимо добуток елементів третьої послідовності, коди яких менші за 40;

**Кінець.**

*Крок 2*

**Початок**

size = 10;

first\_array[size], second\_array[size], third\_array[size];

Ініціалізуємо першу послідовність;

Ініціалізуємо другу послідовність;

Ініціалізуємо третю послідовність;

Обчислимо добуток елементів третьої послідовності, коди яких менші за 40;

**Кінець.**

*Крок 3*

**Початок**

size = 10;

first\_array[size], second\_array[size], third\_array[size];

**initialize\_first\_array**( first\_array[], int array\_size);

Ініціалізуємо другу послідовність;

Ініціалізуємо третю послідовність;

Обчислимо добуток елементів третьої послідовності, коди яких менші за 40;

**Кінець.**

*Крок 4*

**Початок**

size = 10;

first\_array[size], second\_array[size], third\_array[size];

**initialize\_first\_array**( first\_array[], int array\_size);

**initialize\_second\_array**( first\_array[], int array\_size);

Ініціалізуємо третю послідовність;

Обчислимо добуток елементів третьої послідовності, коди яких менші за 40;

**Кінець.**

*Крок 5*

**Початок**

size = 10;

first\_array[size], second\_array[size], third\_array[size];

**initialize\_first\_array**( first\_array[], array\_size);

**initialize\_second\_array**( second\_array[], array\_size);

**initialize\_third\_array**(first\_array[],second\_array[],third\_array[], array\_size);

Обчислимо добуток елементів третьої послідовності, коди яких менші за 40;

**Кінець.**

*Крок 6*

**Початок**

size = 10;

first\_array[size], second\_array[size], third\_array[size];

**initialize\_first\_array**( first\_array[], array\_size);

**initialize\_second\_array**( second\_array[], array\_size);

**initialize\_third\_array**(first\_array[],second\_array[],third\_array[], array\_size);

result = **product**(third\_array[], int third\_array\_size);

**Кінець.**

*Підпрограмми:*

1.

**initialize\_first\_array**( some\_array[], array\_size)

**Початок**

**Повторити** **для** i **від** 1 **до** array\_size

some\_array[i] = 5 \* i + 30;

**Все повторити**

**Кінець.**

2.

**initialize\_second\_array**( some\_array[], array\_size)

**Початок**

**Повторити** **для** i **від** 1 **до** array\_size

some\_array[i] = 60 - 5\*i;

**Все повторити**

**Кінець.**

3.

**initialize\_third\_array**(first\_array[], second\_array[], result\_array[], array\_size)

**Початок**

result\_array\_size = 0;

**Повторити** **для** i **від** 1 **до** array\_size

**Повторити** **для** j **від** 1 **до** array\_size

**Якщо**(first\_array[i] == second\_array[j])

result\_array[result\_array\_size] = first\_array[i];

result\_array\_size++;

**Все якщо;**

**Все повторити;**

**Все повторити;**

**Кінець.**

4.

**product** (some\_array[], some\_array\_size)

**Початок**

**Повторити** **для** i **від** 1 **до** array\_size

**Якщо (**relevant\_value == last\_value)

**То**

result \*= some\_array[i];

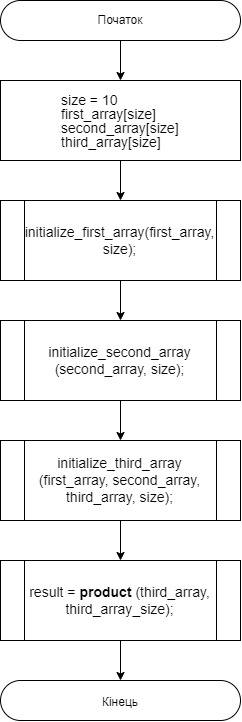
**Все якщо;**

**Все повторити;**

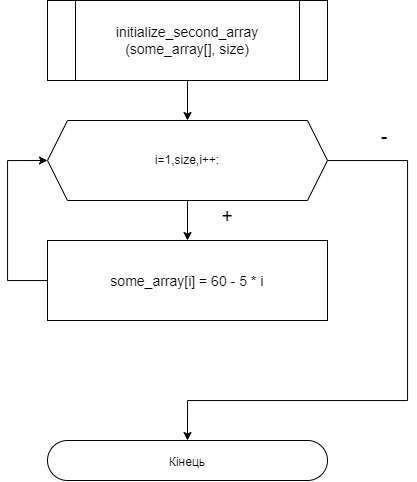
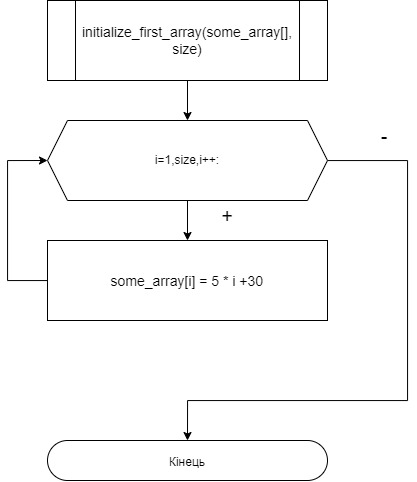
**Повернути** result;

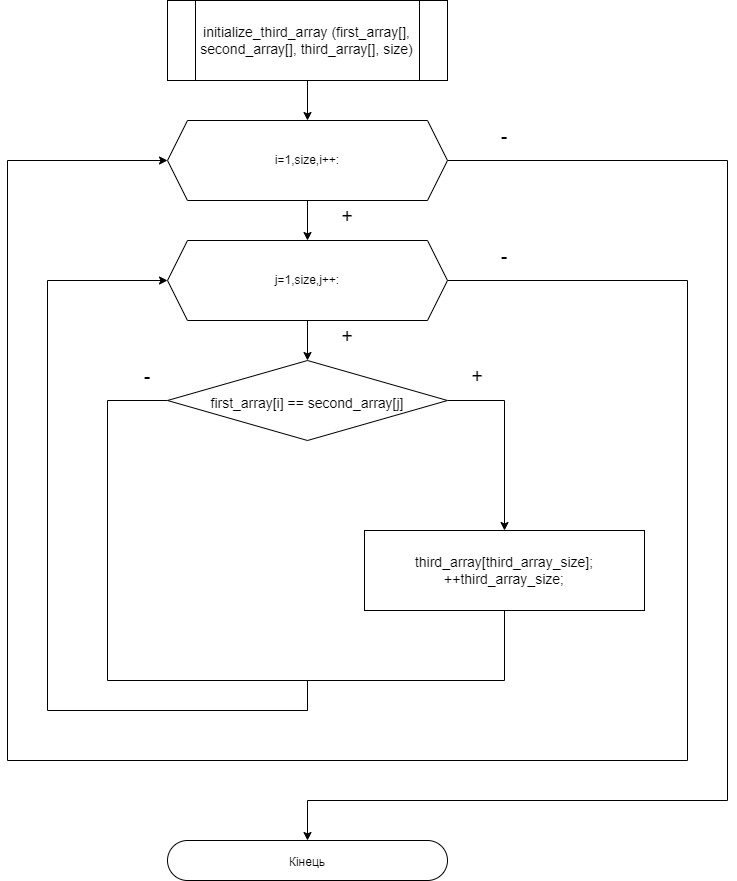
**Кінець.**

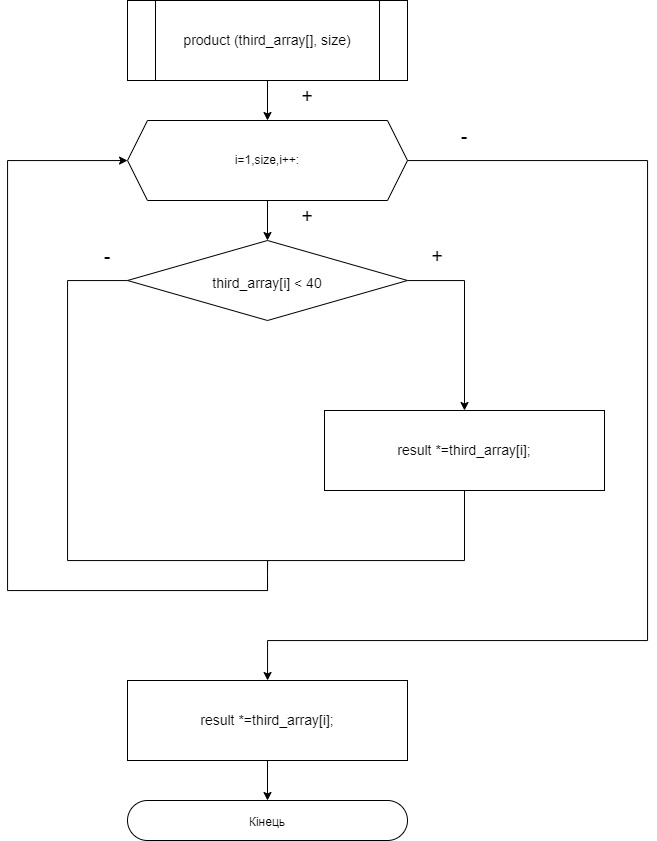
*Блок-схема:*

****

*Підпрограми:*



****

****

**Випробування алгоритму:**

|  |  |
| --- | --- |
| **Блок** | **Дія** |
|  | Початок |
| **1.** | Формується перша послідовність символів (▲, #, (, -, 2, 7, <, A, F, K), коди яких: ( 30, 35, 40, 45, 50, 55, 60, 65, 70, 75) відповідно. |
| **2.** | Формується друга послідовність символів (<, 7, 2, -, (, #,▲, ↓, ¶, \*), коди яких: ( 60, 55, 50, 45, 40, 35, 30, 25, 20, 15) відповідно. |
| **3.** | Формується третя послідовність символів (<, 7, 2, -, (, #,▲), коди яких: ( 60, 55, 50, 45, 40, 35, 30) відповідно. |
| **4.** | З третьої послідовності обираються символи, коди яких менші за 40: ( #, ▲) та знаходиться їх добуток:  “#” \* “▲” = 35 \* 30 = 1050 = “→” |
|  | Кінець |

**Код программи:**

#include <iostream>

#include <iomanip>

using namespace std;

void initialize\_first\_array(char [], int);

void initialize\_second\_array(char [], int);

int initialize\_third\_array(char [], char [], char [], int);

char product(char[], int);

int main()

{

const int size = 10;

char first\_array[size], second\_array[size], third\_array[size];

int third\_array\_size;

cout << "First array:\n";

initialize\_first\_array(first\_array , size);

cout << "Second array:\n";

initialize\_second\_array(second\_array, size);

cout << "Third array:\n";

third\_array\_size = initialize\_third\_array(first\_array, second\_array, third\_array, size);

char result = product(third\_array, third\_array\_size);

cout << "Result: " << result;

return 1;

}

void initialize\_first\_array(char some\_array[], int array\_size)

{

for (int i = 0; i < array\_size; ++i)

{

some\_array[i] = 5 \* i + 30;

cout << setw(3) << some\_array[i];

}

cout << '\n';

}

void initialize\_second\_array(char some\_array[], int array\_size)

{

for (int i = 0; i < array\_size; ++i)

{

some\_array[i] = 60 - 5 \* i;

cout << setw(3) << some\_array[i];

}

cout << '\n';

}

int initialize\_third\_array(char first\_array[], char second\_array[], char result\_array[], int array\_size)

{

int result\_array\_size = 0;

for (int i = 0; i < array\_size; ++i) //алгоритм пошуку однакових елементів 1 та 2 масиву та додавання їх до 3 массиву

{

for (int j = 0; j < array\_size; ++j)

{

if(first\_array[i] == second\_array[j])

{

result\_array[result\_array\_size] = first\_array[i];

cout << setw(3) << result\_array[result\_array\_size];

result\_array\_size++;

continue;

}

}

}

cout << '\n';

return result\_array\_size;

}

char product(char some\_array[], int some\_array\_size)

{

char product = some\_array[0];

for (int i = 1; i < some\_array\_size; ++i)

{

if (some\_array[i] < 40)

{

product \*= some\_array[i];

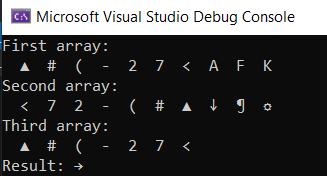
}

}

return product;

}

*Копії екранних форм:*

**

**Висновок:** В ході лабораторної роботи я дослідив методи лінійного пошуку у впорядкованих послідовностях, навчився використовувати їх для виконання задач та зображати у вигляді блок-схеми та псевдокоду.