

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 2

Виконав студент ПІ-15, Богун Даниїл Олександрович
(шифр, прізвище, ім'я, по батькові)

Перевірила Вечерковська 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 неупорядкованих послідовностях та набутти практичних навичок їх використання під час складання програмних специфікацій.

Варіант 2

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.

2	$5 * i + 30$	$60 - 5 * i$	Добуток елементів, коди яких менше 40
---	--------------	--------------	---------------------------------------

Постановка задачі

Створюємо два масиви, заповнюємо їх елементами, які задаються формулами $5 * i + 30$ (перший масив) та $60 - 5 * i$ (другий масив). Третій масив заповнюємо спільними елементами першого і другого масивів. Далі працюємо з третім масивом – знаходимо елементи, коди яких менше ніж 40, і множимо їх. Їх добуток – результат роботи програми.

Змінна	Тип	Ім'я	Призначення
Перший масив	символьний	a	Результат, проміжні дані
Другий масив	символьний	b	Результат, проміжні дані
Третій масив	символьний	c	Результат, проміжні дані
Розмір масивів a та b	Цілий	SIZE	Початкове дане
Лічильник елементів масиву c	Цілий	k	Проміжне дане
Лічильник елементів масивів a та b	Цілий	i	Параметр циклу
Добуток елементів масиву	символьний	m	Проміжне дане
Генерація масивів a та b	Процедура	arr_a_b	Початкове дане
Генерація масиву c	Процедура	arr_c	Початкове дане
Виведення трьох масивів	Процедура	prinArrs	Результат
Знаходження добутку елементів, коди яких менше 40	Процедура	dobutok	Результат

Спочатку створюємо два масиви a і b . Розмір масивів – 10 символів, тип масивів – `char`. Коди елементів першого масиву розраховуються за формулою $5 * i + 30$. Коди елементів другого масиву розраховуються за формулою $60 - 5 * i$. i приймає значення від 0 до 9. Ці дії виконуємо всередині функції `arr_a_b`.

У функції `arr_c` знаходимо спільні елементи масивів a та b . Вводимо лічильник k . Вводимо вкладений цикл, який буде порівнювати елемент першого масиву зі всіма елементами другого і буде збільшуватися на 1, коли елемент 1 масиву було порівняно з усіма елементами другого масиву. Якщо i -тий елемент першого масиву рівний j -тому елементу другого масиву, то k -тому елементу третього масиву присвоюємо значення i -того елементу другого масиву. k збільшуємо на 1. Цикл закінчується, коли i досягає максимального значення (9).

У функції `prinArgs` ми за допомогою циклу `for` (i – лічильник) виводимо значення масивів на консоль. Початкове значення i – 0, з кожною ітерацією збільшуємо i на 1, цикл продовжуємо, поки i не досягне максимального значення (9).

У функції `dobutok` ми за допомогою циклу `for` (i – лічильник) знаходимо значення m . Початкове значення $i = 1$. Початкове значення $m = 0$. Якщо код елемента масиву менше 40, m – добуток цього i попереднього елементів.

Виводимо на консоль значення добутку.

Розв'язання

Крок 1. Визначемо основні дії.

Крок 2. Деталізуємо дію генерації двох масивів за допомогою підпрограми.

Крок 3. Деталізуємо дію знаходження елементів третього масиву за допомогою підпрограми.

Крок 4. Деталізуємо дію виведення елементів масивів.

Крок 5. Деталізуємо дію знаходження добутку за допомогою підпрограми.

Крок 6. Вивід добутку.

Псевдокод алгоритму

Основна програма:

Початок

SIZE = 10

arr_a_b(a, b, SIZE)

k = arr_c(a, b, c, SIZE)

prinArrs(a, b, c, k, SIZE)

res = dobutok(c, k)

Вивести res

Кінець

Підпрограми :

arr_a_b (a, b,SIZE)

Повторити

для i від 0 до SIZE

a[i] = 5 * i + 30;
b[i] = 60 - 5 * i;

Все повторити

Кінець

arr_c (a, b, c, SIZE)

k = 0

Повторити

для i від 0 до SIZE

Повторити

для j від 0 до SIZE

Якщо a[i] == b[j]

то c[k] = a[i]

k = k + 1;

Все повторити

Повернути k

Все повторити

Кінець

prinArrs (a, b, c, t, SIZE)

Вивід “Array 1 Array 2 Array 3”

Повторити

для i від 0 до SIZE

Вивід “a[i] b[i] c[i]”

Все повторити

Кінець

dobutok(a, t)

$m = 1$

Повторити

для i від 0 до t

Якщо $c[i] < 40$

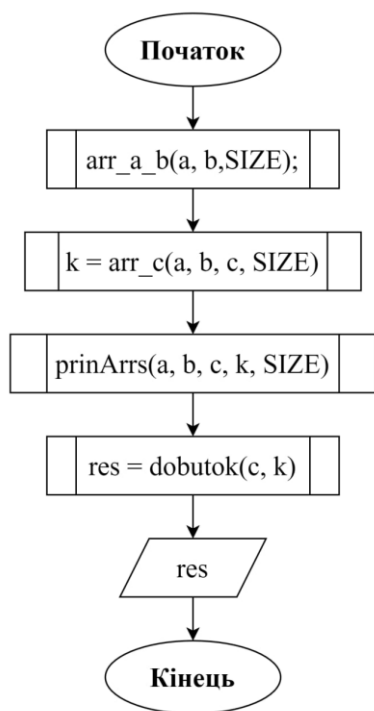
$m = c[i] * m$

Все повторити

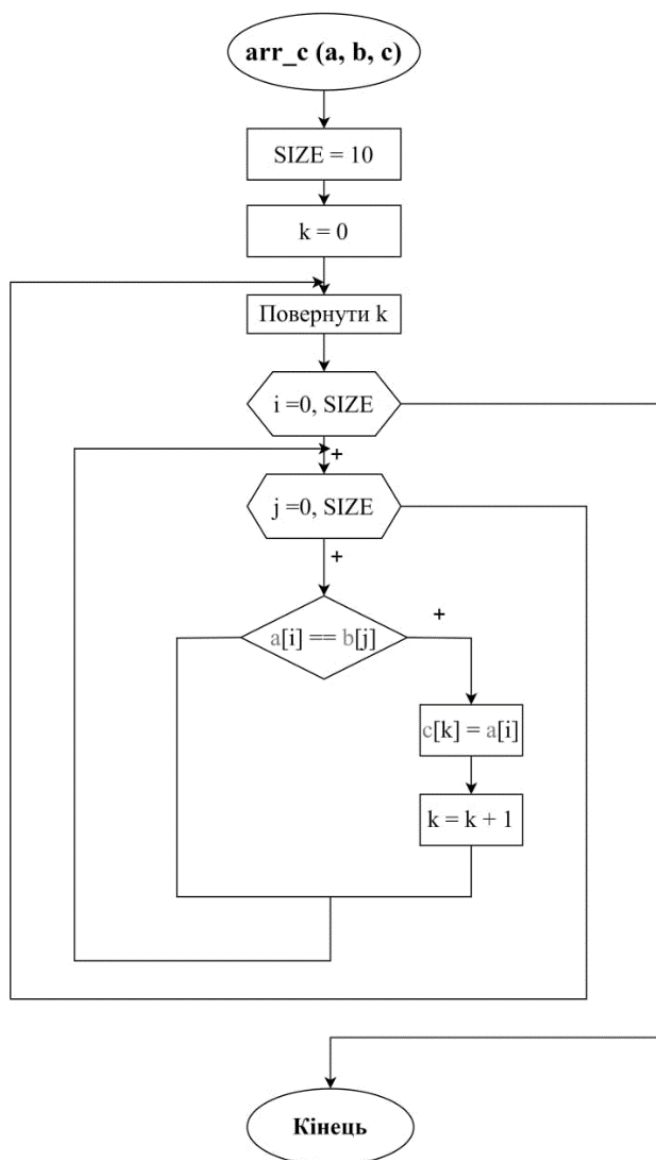
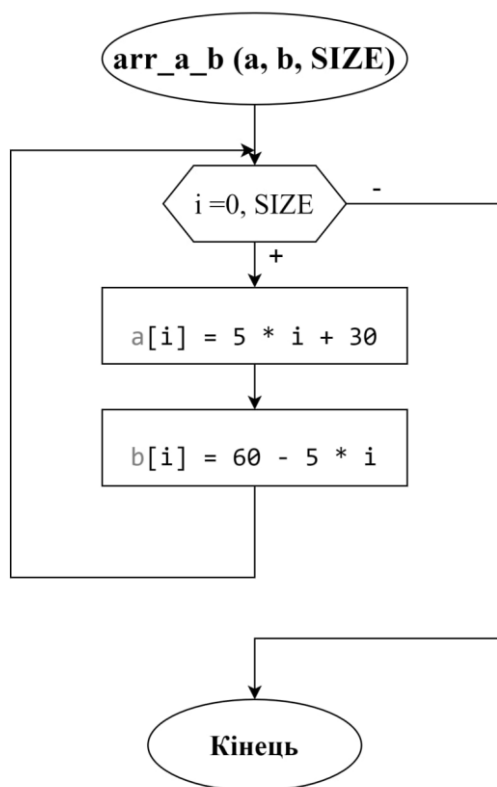
повернути m

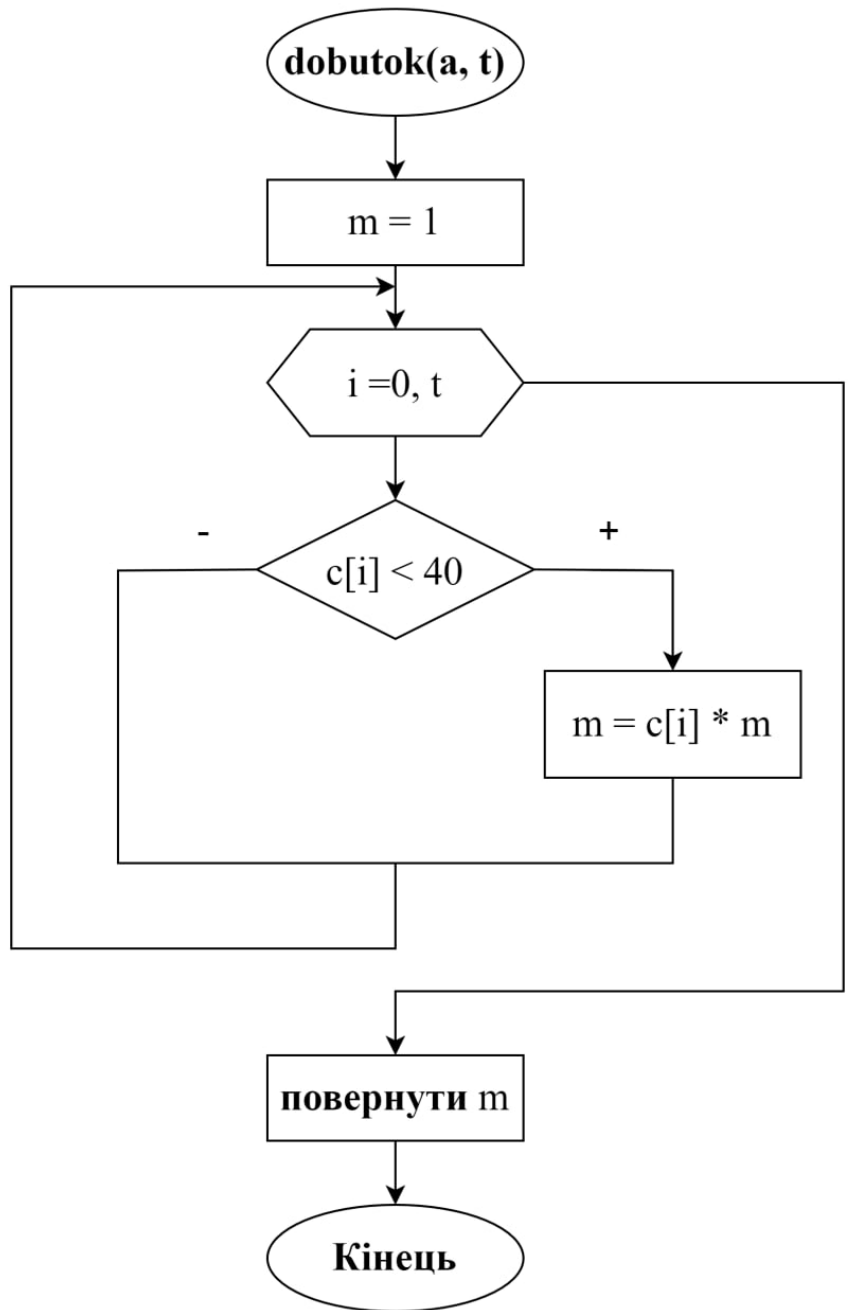
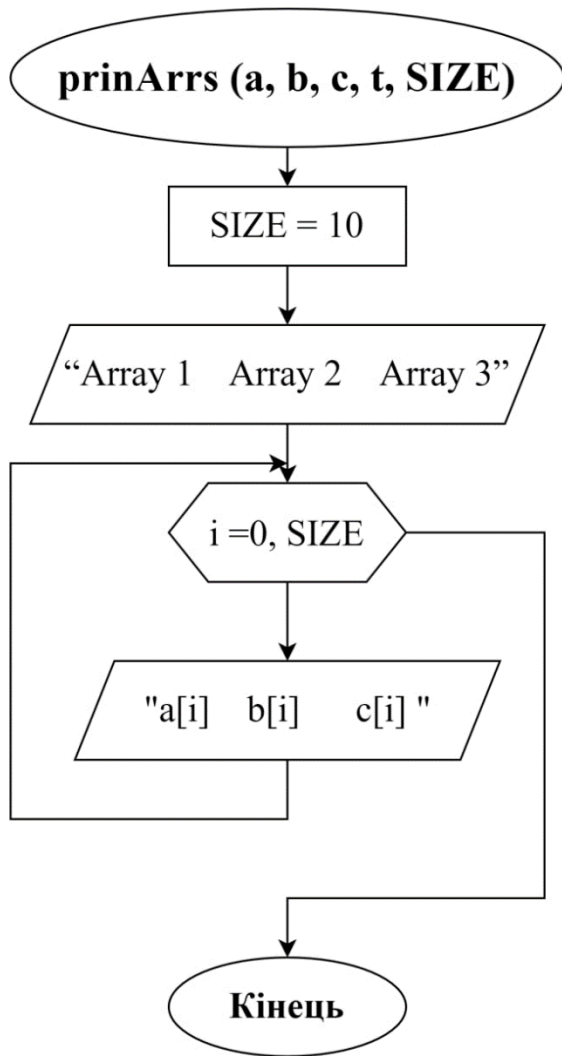
Блок-схема алгоритму

Основна програма:



Підпрограми:





Код:

```
#include <iostream>
#include <windows.h>

int arr_c(char a[], char b[], char c[], int SIZE);
void arr_a_b(char a[], char b[], int SIZE);
void prinArrs(char a[], char b[], char c[], int k, int SIZE);
char dobutok(char c[], int t);
```

```
int main() {
    const int SIZE = 10;
    char a[SIZE], b[SIZE], c[SIZE] = { 0 };
    arr_a_b(a, b, SIZE);
    int k = arr_c(a, b, c, SIZE);
    prinArrs(a, b, c, k, SIZE);
    char res = dobutok(c, k);
    std::cout << "dobutok = " << res;
}
```

```
void arr_a_b(char a[], char b[], int SIZE) {
    for (int i = 0; i < SIZE; i++) {
        a[i] = 5 * i + 30;
        b[i] = 60 - 5 * i;
    }
}
```

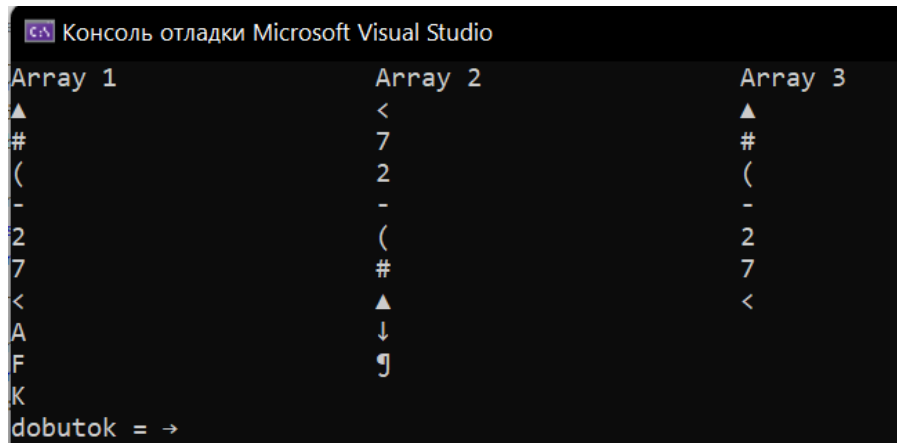
```
int arr_c(char a[], char b[], char c[], int SIZE) {
    int k = 0;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (a[i] == b[j]) {
                c[k] = a[i];
                k++;
                break;
            }
        }
    }
    return k;
}
```

```
void prinArrs(char a[], char b[], char c[], int t, int SIZE) {
    std::cout << "Array 1" << "\t\t" << "Array 2" << "\t\t" << "Array 3" << "\n";
    for (int i = 0; i < SIZE; i++) {
        std::cout << a[i] << "\t\t" << b[i] << "\t\t" << c[i] << "\n";
    }
}
```

```
void prinArrs(char a[], char b[], char c[], int t, int SIZE) {
    std::cout << "Array 1" << "\t\t" << "Array 2" << "\t\t" << "Array 3" << "\n";
    for (int i = 0; i < SIZE; i++) {
        std::cout << a[i] << "\t\t" << b[i] << "\t\t" << c[i] << "\n";
    }
}
```

```
char dobutok(char c[], int t) {
    char m = 1;
    for (int i = 0; i < t; i++) {
        if (c[i] < 40)
            m = c[i] * m;
    }
    return m;
}
```


Тестування алгоритму:



The screenshot shows the 'Консоль отладки Microsoft Visual Studio' (Visual Studio Debug Console) with three columns of data labeled 'Array 1', 'Array 2', and 'Array 3'. Each column contains a sequence of characters: a triangle, a hash, an opening parenthesis, a hyphen, the number 2, the number 7, a less-than sign, the letter A, the letter F, and the letter K. At the bottom, the text 'dobutok = →' is visible.

Array 1	Array 2	Array 3
▲	<	▲
#	7	#
(2	(
-	-	-
2	(2
7	#	7
<	▲	<
A	↓	
F	↵	
K		

dobutok = →

Висновки

Ми дослідили алгоритми лінійного пошуку в масивах та набули практичних навичок їх створення та використання під час складання програмних специфікацій.