

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 32

Виконав студент ІП-14, Шляхтун Денис Михайлович
(шифр, прізвище, ім'я, по батькові)

Перевірів доц. кафедри ІІІ Мартинова Оксана Петрівна
(прізвище, ім'я, по батькові)

Лабораторна робота 7

Дослідження лінійного пошуку в послідовностях

Мета – дослідити методи послідовного пошуку у впорядкованих і неупорядкованих послідовностях та набуті практичних навичок їх використання під час складання програмних специфікацій.

Задача:

1. Описати три змінні індексованого типу з 10 символьних значень.
2. Ініціювати дві змінні виразами:
 - a. $74 - i$
 - b. $65 + 2 * i$
3. Ініціювати третю змінну рівними значеннями двох попередніх змінних.
4. Знайти кількість елементів, коди яких менше 67

Постановка задачі. Результатом розв'язку є кількість елементів з кодами менше 67 масиву, що складається з однакових елементів двох масивів, значення яких задаються формулами із умови. Ввідних даних не вимагається.

Побудова математичної моделі. Складемо таблицю імен змінних

<i>Змінна</i>	<i>Тип</i>	<i>Ім'я</i>	<i>Призначення</i>
<i>Основні змінні</i>			
Перший масив	Символьний	A[10]	Проміжне дане
Другий масив	Символьний	B[10]	Проміжне дане
Масив спільних елементів	Символьний	C[10]	Проміжне дане
Кількість елементів у масиві C	Цілий	k	Проміжне дане
Кількість елементів	Цілий	n	Результат
<i>Змінні, що використовуються у підпрограмах</i>			
Масив	Символьний	arr[]	Проміжне дане
Доданок із заданої формули	Цілий	add	Проміжне дане
Множник із заданої формули	Цілий	mult	Проміжне дане
Лічильники	Цілий	i, l	Проміжне дане

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію визначення перших двох масивів.

Крок 3. Деталізуємо дію визначення третього масиву.

Крок 4. Деталізуємо дію визначення кількості елементів менше 67.

Псевдокод.

Крок 1.

початок

знаходження першого та другого масиву

знаходження третього масиву

знаходження кількості елементів

кінець

Крок 2.

початок

A = arrayInit(A, 74, -1)

B = arrayInit(B, 65, 2)

знаходження третього масиву

знаходження кількості елементів

кінець

Крок 3.

початок

A = arrayInit(A, 74, -1)

B = arrayInit(B, 65, 2)

C, k = arrayEqual(A, B, C)

знаходження кількості елементів

кінець

Крок 4.

початок

A = arrayInit(A, 74, -1)

B = arrayInit(B, 65, 2)

C, k = arrayEqual(A, B, C)

n = arrayCheck(C, k)

виведення n

кінець

функція arrayInit(arr[], add, mult)

повторити для $i = 0; i < 10; i++$

arr[i] = add + mult * i

все повторити

arrayOutput(arr, 10)

повернути arr

кінець функції

функція arrayOutput(arr[], k)

повторити для $i = 0; i < k; i++$

виведення arr[i]

все повторити

кінець функції

функція arrayEqual(A[], B[], C[])

k = 0

повторити для $i = 0; i < k; i++$

повторити для $l = 0; l < k; l++$

якщо A[i] = B[l]

$C[k]=A[i]$

$k++$

все якщо

все повторити

все повторити

arrayOutput(C, k)

повернути C, k

кінець функції

функція arrayCheck(C[], k)

$n = 0$

повторити для $i = 0; i < k; i++$

якщо $C[i] < 67$

$n++$

виведення C[i]

все якщо

все повторити

повернути n

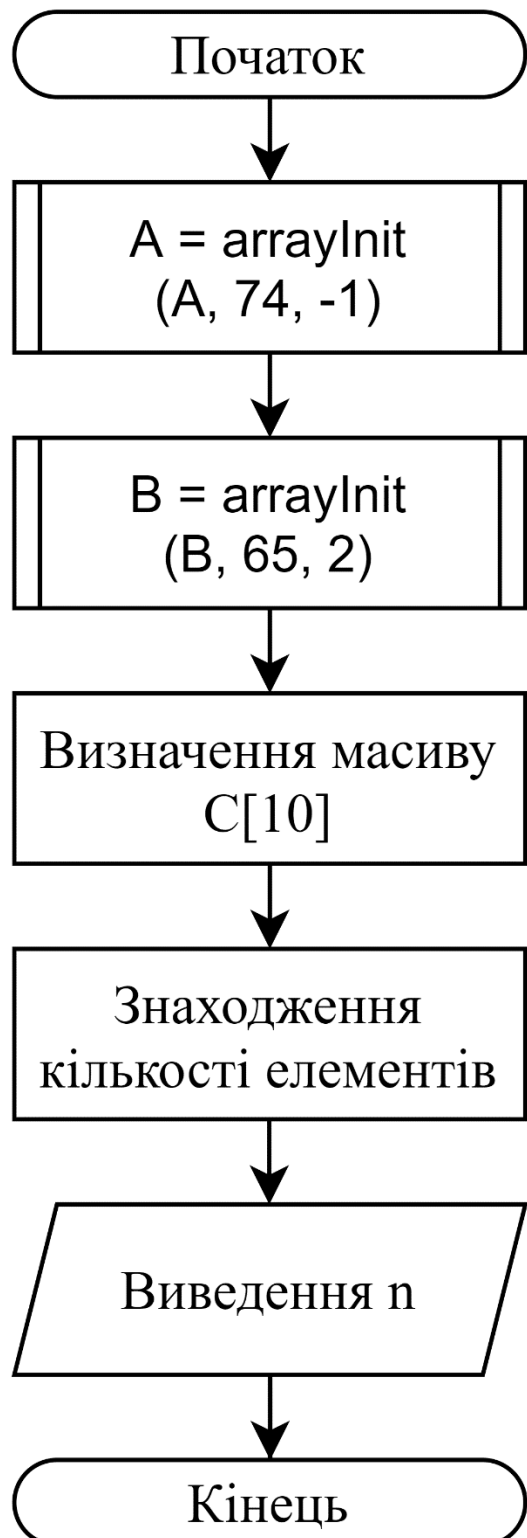
кінець функції

Блок-схема алгоритму

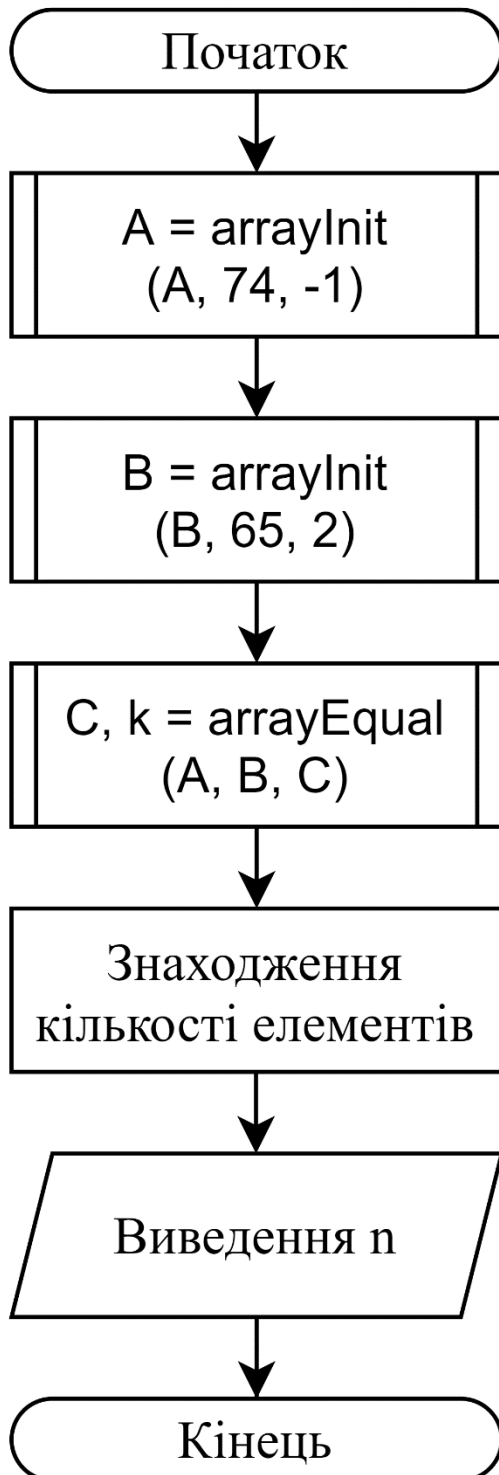
Крок 1



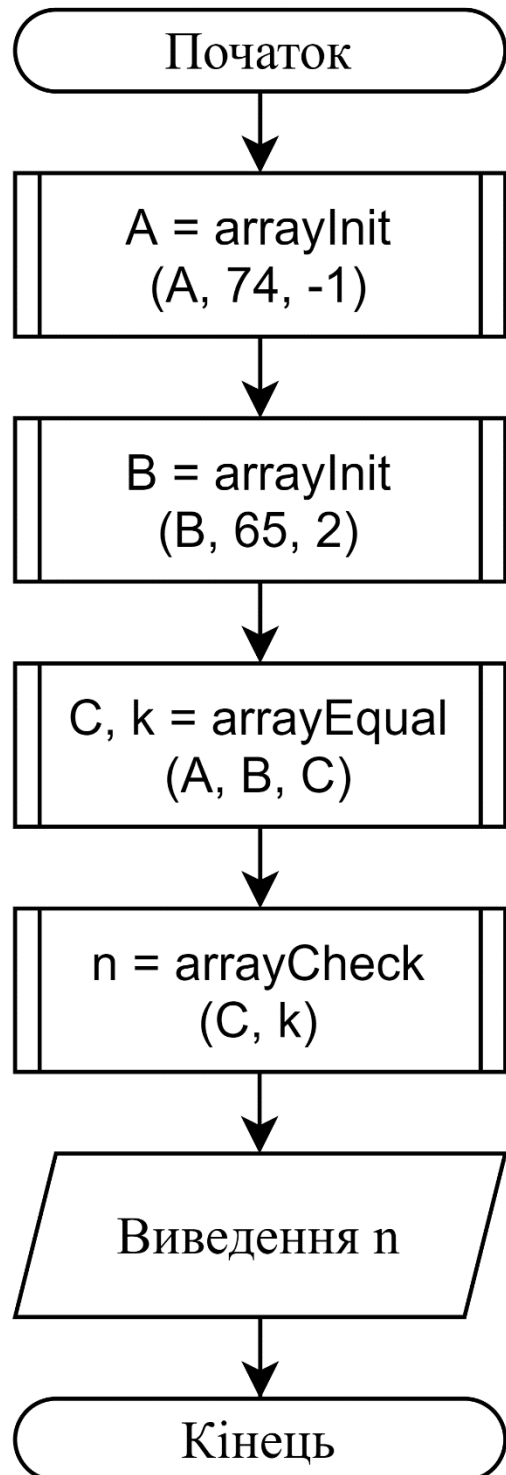
Крок 2

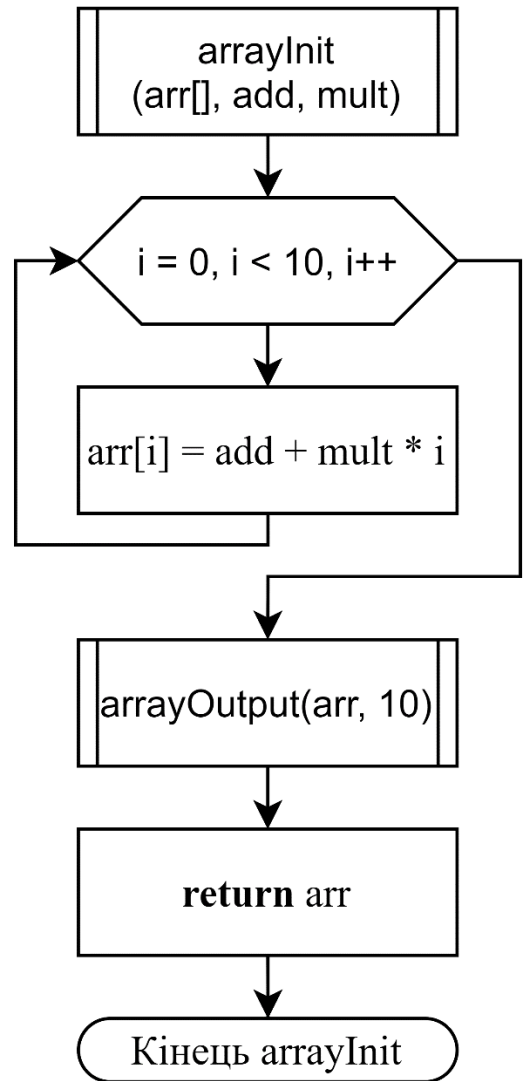
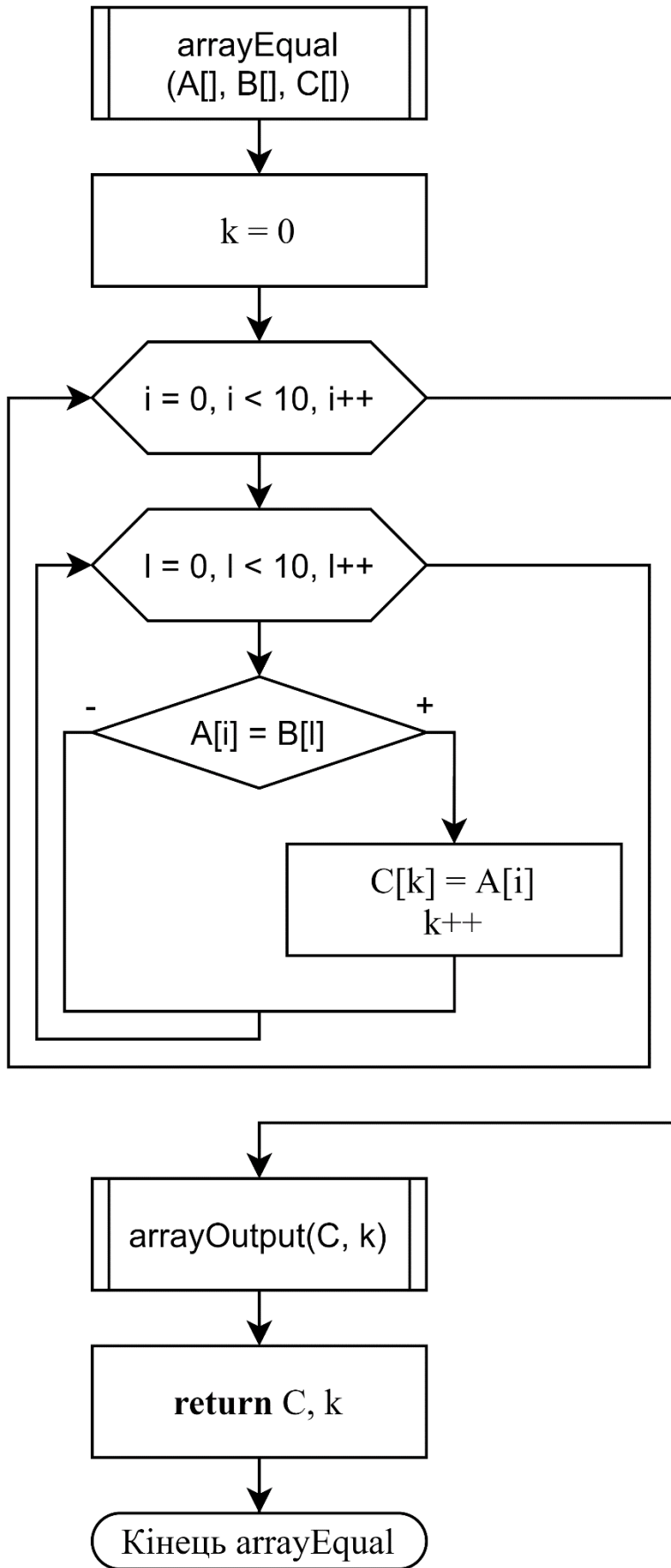


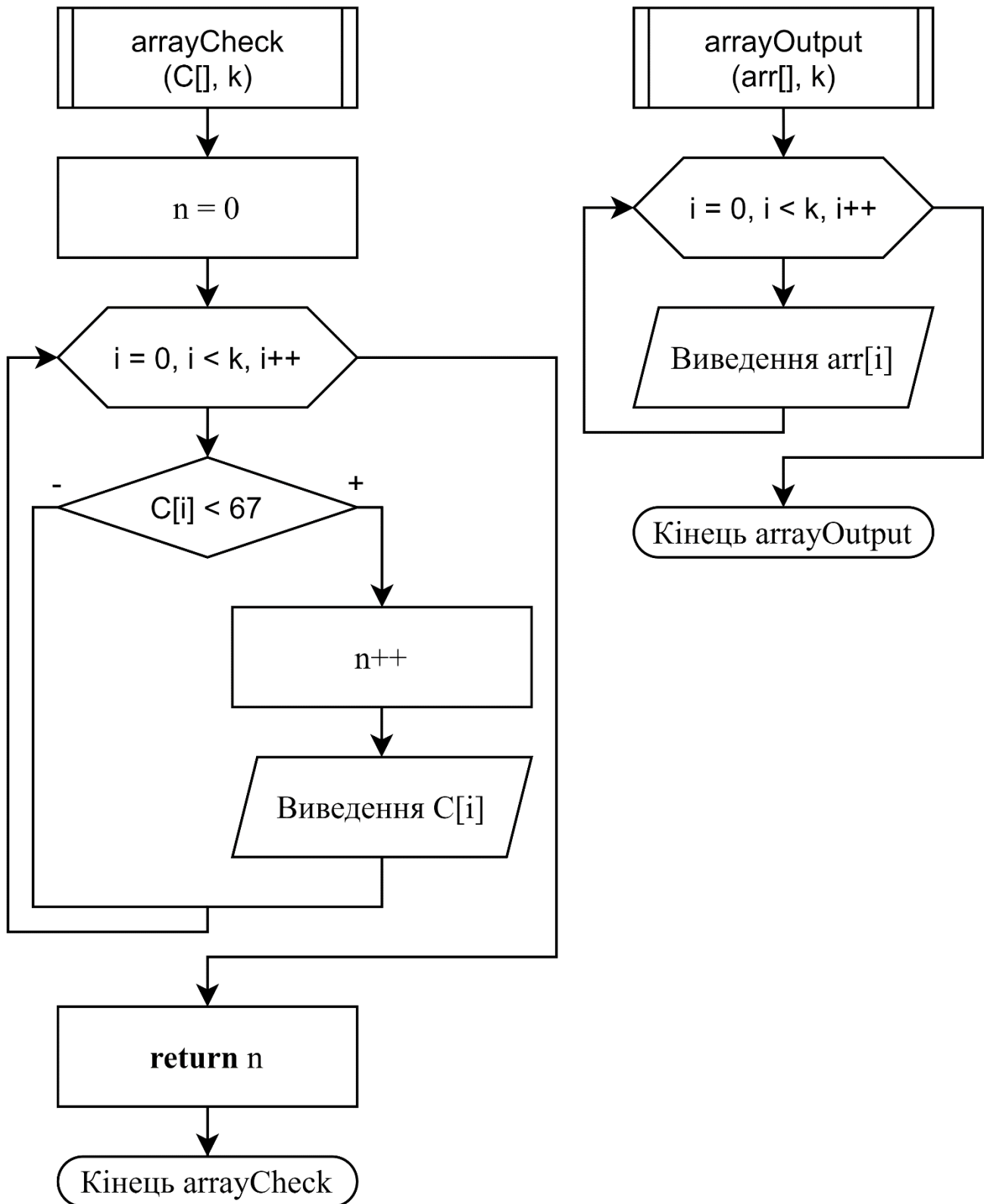
Крок 3



Крок 4







Код програми (C++).

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  char* arrayInit(char arr[], int add, int mult);
6  int arrayEqual(char A[], char B[], char C[]);
7  int arrayCheck(char C[], int k);
8  void arrayOutput(char arr[], int k);
9
10 int main() {
11     //визначення першого масиву
12     char* A = new char[10];
13     cout << "A: ";
14     A = arrayInit(A, 74, -1);
15
16     //визначення другого масиву
17     char* B = new char[10];
18     cout << endl << "B: ";
19     B = arrayInit(B, 65, 2);
20
21     //визначення третього масиву
22     char C[10];
23     cout << endl << "C: ";
24     int k = arrayEqual(A, B, C);
25
26     //знаходження елементів менше 67
27     cout << endl << "Elements under 67: ";
28     int n = arrayCheck(C, k);
29     cout << endl << "n: " << setw(3) << n << endl;
30
31     system("pause");
32 }
```

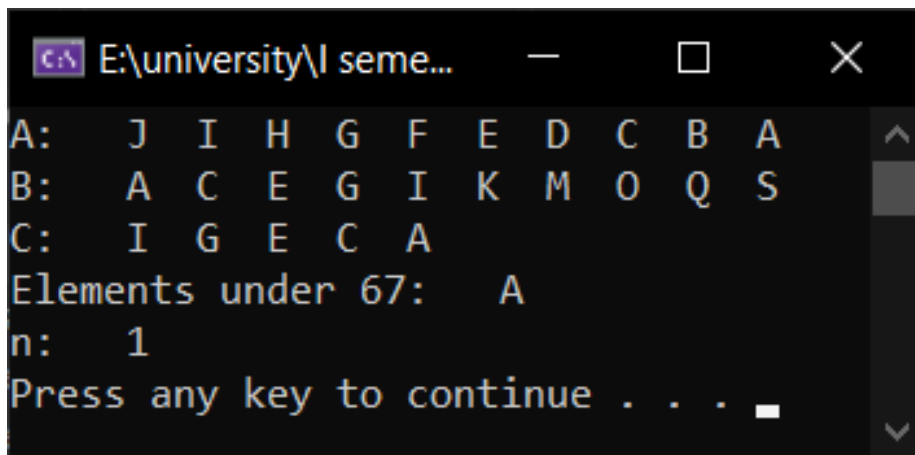
```
34 char* arrayInit(char arr[], int add, int mult)
35 {
36     for (int i = 0; i < 10; i++)
37     {
38         arr[i] = add + mult * i;
39     }
40     arrayOutput(arr, 10);
41     return arr;
42 }
43
44 int arrayEqual(char A[], char B[], char C[])
45 {
46     int k = 0;
47     for (int i = 0; i < 10; i++)
48     {
49         for (int l = 0; l < 10; l++)
50         {
51             if (A[i] == B[l])
52             {
53                 C[k] = A[i];
54                 k++;
55             }
56         }
57     }
58     arrayOutput(C, k);
59     return k;
60 }
```

```

61 int arrayCheck(char c[], int k)
62 {
63     int n = 0;
64     for (int i = 0; i < k; i++)
65     {
66         if (c[i] < 67)
67         {
68             n++;
69             cout << setw(3) << c[i];
70         }
71     }
72     return n;
73 }
74
75 void arrayOutput(char arr[], int k)
76 {
77     for (int i = 0; i < k; i++)
78     {
79         cout << setw(3) << arr[i];
80     }
81 }

```

Результат виконання програми.



```

E:\university\I seme...
A:  J  I  H  G  F  E  D  C  B  A
B:  A  C  E  G  I  K  M  O  Q  S
C:  I  G  E  C  A
Elements under 67:  A
n:  1
Press any key to continue . . . .

```

Перевірка.

Масив А:

- $i = 0, A[0] = 74 - 0 = 74, \text{ASCII} - J$
- $i = 1, A[1] = 74 - 1 = 73, \text{ASCII} - I$

- $i = 2, A[2] = 74 - 2 = 72$, ASCII – Н
- $i = 9, A[9] = 74 - 9 = 65$, ASCII – А

Масив В:

- $i = 0, B[0] = 65 + 2 * 0 = 65$, ASCII – А
- $i = 1, B[0] = 65 + 2 * 1 = 67$, ASCII – С
- $i = 2, B[0] = 65 + 2 * 2 = 69$, ASCII – Е

Єдиний спільний елемент з кодом менше 67 – А, тож такий елемент один

Висновок.

При виконанні лабораторної роботи було використано лінійний пошук – послідовний пошук даних, який виконується за допомогою оператора повторення з укладеним умовним оператором. Даний пошук використовувався над послідовностями значень або масивами, що розглядається як іменована сукупність значень одного типу, а кожне значення має свій індекс.