Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені І. Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт
з лабораторної роботи № 7
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження лінійного пошуку в послідовностях»

Варіант 2

Виконав студент: ІП-13 Бабашев Олексій Дмитрович

Перевірив: Вечерковська Анастасія Сергіївна

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 2

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.

3. Ініціювання третьої змінної рівними значеннями двох попередніх змінних.

4. Обробки третьої змінної згідно з варіантом.

1ий масив 5 * i + 30 2ий масив 60 - 5 * i. Знайти: добуток елементів, коди яких менше 40

1) Постановка задачі:

Сформувати третій масив з однакових елементів третього та другого масиву, використавши алгоритм лінійного пошуку. Обчислити добуток елементів третього масиву код яких менший за 40.

2) Побудова математичної моделі:

Змінна	Тип	Назва	Призначення
Перший масив	Символьний масив	arr1	Вихідне дане
Другий масив	Символьний масив	arr2	Вихідне дане
Третій масив	Символьний масив	arr3	Вихідне дане
Розмір другого на першого масивів(const)	Цілий	arr12size	Проміжне дане
Розмір третього масиву	Цілий	arr3size	Проміжне дане
Результат добутку елементів третього масиву код яких менше 40	Символ	res	Вихідне дане
Параметричний розмір масиву в підпрограмі	Цілий	arrsize	Параметр підпрограми
Параметричний масив для підпрограми	Символьний масив	farr	Параметр підпрограми

Параметричний масив для підпрограми	Символьний масив	sarr	Параметр підпрограми
Параметричний масив для підпрограми	Символьний масив	taar	Параметр підпрограми
Символьний добуток	Символьний	product	Проміжне дане
Лічильник	Цілий	i	Проміжне дане
Лічильник	Цілий	j	Проміжне дане

Математичне формулювання задачі зводиться до ініціалізації перший двох масивів за допомогою initialize_arr1(2)(f(s)arr[], arrsize), де f(s)arr перший та другий масив, а arrsize їх розмір.

Далі ініціалізуємо третій масив, який складається з однакових елементів двох інших, за допомогою функції initialize_arr3(farr[], sarr[], tarr[], arrsize), де farr, sarr, tarr перший другий та третій масиви відповідно, arrsize – розмір масивів. Розмір третього масиву визначається у цій функції та повертається як arr3size.

Обчислити символьний добуток елементів третього масиву код яких менший за 40, використовуючи функцію product(tarr[], arrsize) tarr – масив, arrsize – розмір масива. Повертаємо символьний добуток product та присвоюємо його значення параметру res.

Розв’язання:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо ініціалізацію двох масивів.

Крок 3. Деталізуємо ініціалізацію третього масиву

Крок 4. Деталізуємо обчислення добутку елементів символьного масиву, менших за 40.

Крок 5. Деталізуємо підпрограми.

3) Псевдокод:

Крок 1

Початок

arr12size = 10

ініціалізація двох масивів.

ініціалізація третього масиву

обчислення добутку елементів

Вивести res

Кінець

Крок 2

Початок

arr12size = 10

initialize_arr1(arr1, arr12size)

ініціалізація третього масиву

обчислення добутку елементів

Вивести res

Кінець

Крок 3

Початок

arr12size = 10

initialize_arr1(arr1, arr12size)

initialize_arr3(arr1, arr2, arr3, arr12size)

обчислення добутку елементів

Вивести res

Кінець

Крок 4

Початок

arr12size = 10

initialize_arr1(arr1, arr12size)

initialize_arr3(arr1, arr2, arr3, arr12size)

res = product(arr3, arr3size)

Вивести res

Кінець

Крок 5

Підпрограми:

initialize_arr1(farr[], arrsize)

Початок

Повторити для i від 0 до i<arrsize з кроком i++

farr[i] = 5 * i + 30

Вивести farr[i]

Все повторити

Кінець

initialize_arr2(sarr[], arrsize)

Початок

Повторити для i від 0 до i<arrsize з кроком i++

sarr[i] = 60 - 5*i

Вивести sarr[i]

Все повторити

Кінець

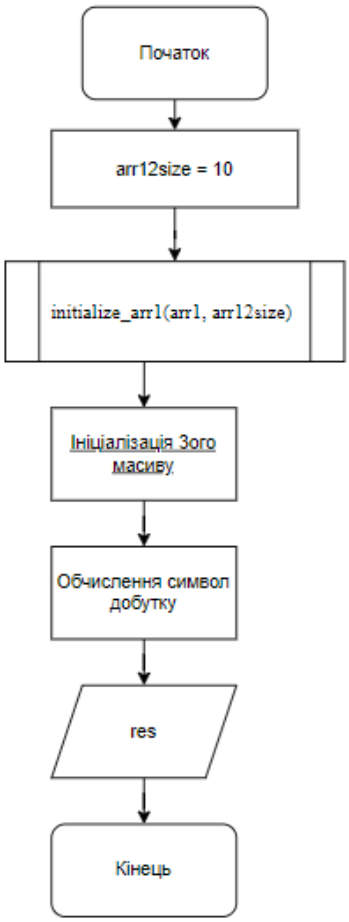
initialize_arr3(farr[],sarr[],tarr[], arrsize)
Початок
arr3size = 0
Повторити для i від 0 до i<arrsize з кроком i++
Повторити для j від 0 до j<arrsize з кроком j++
Якщо farr[i] == sarr[j]
То tarr[arr3size] = farr[i]
Вивести tarr[arr3size]
arr3size++
Все якщо
Все повторити
Все повторити
Повернути arr3size
Кінець.

product(tarr[], arrsize **)**
Початок
product = 1
Повторити для i від 0 до i<arrsize з кроком i++
Якщо tarr[i]<40
То product *= tarr[i];
Все якщо
Все повторити
Повернути product
Кінець.

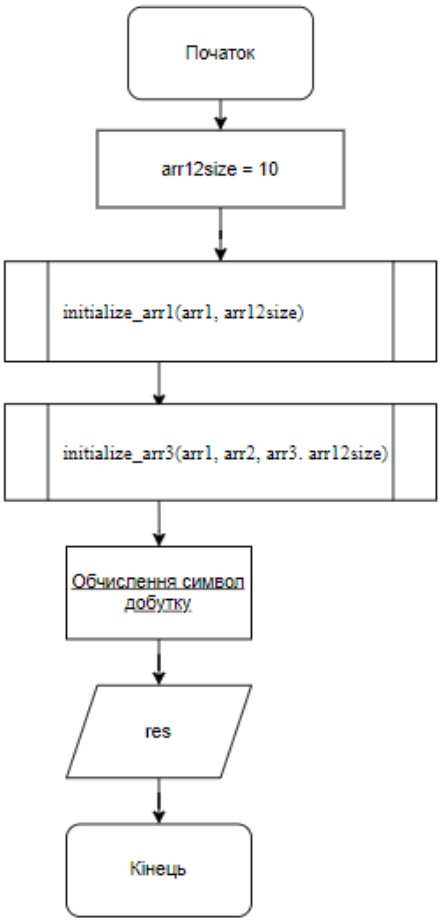
4) Блок-схема:

Крок 1

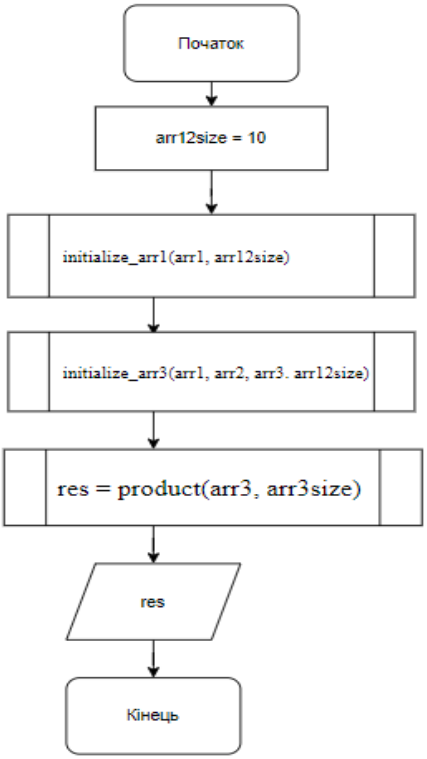
Крок 2



Крок 3



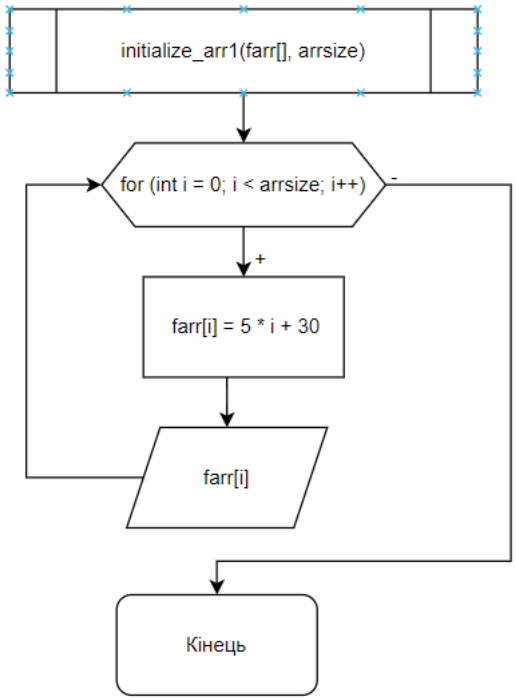
Крок 4



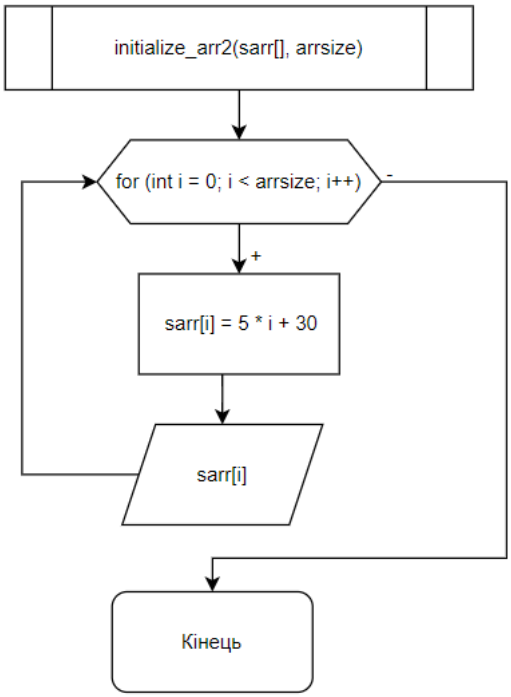
Крок 5

Підпрограми:

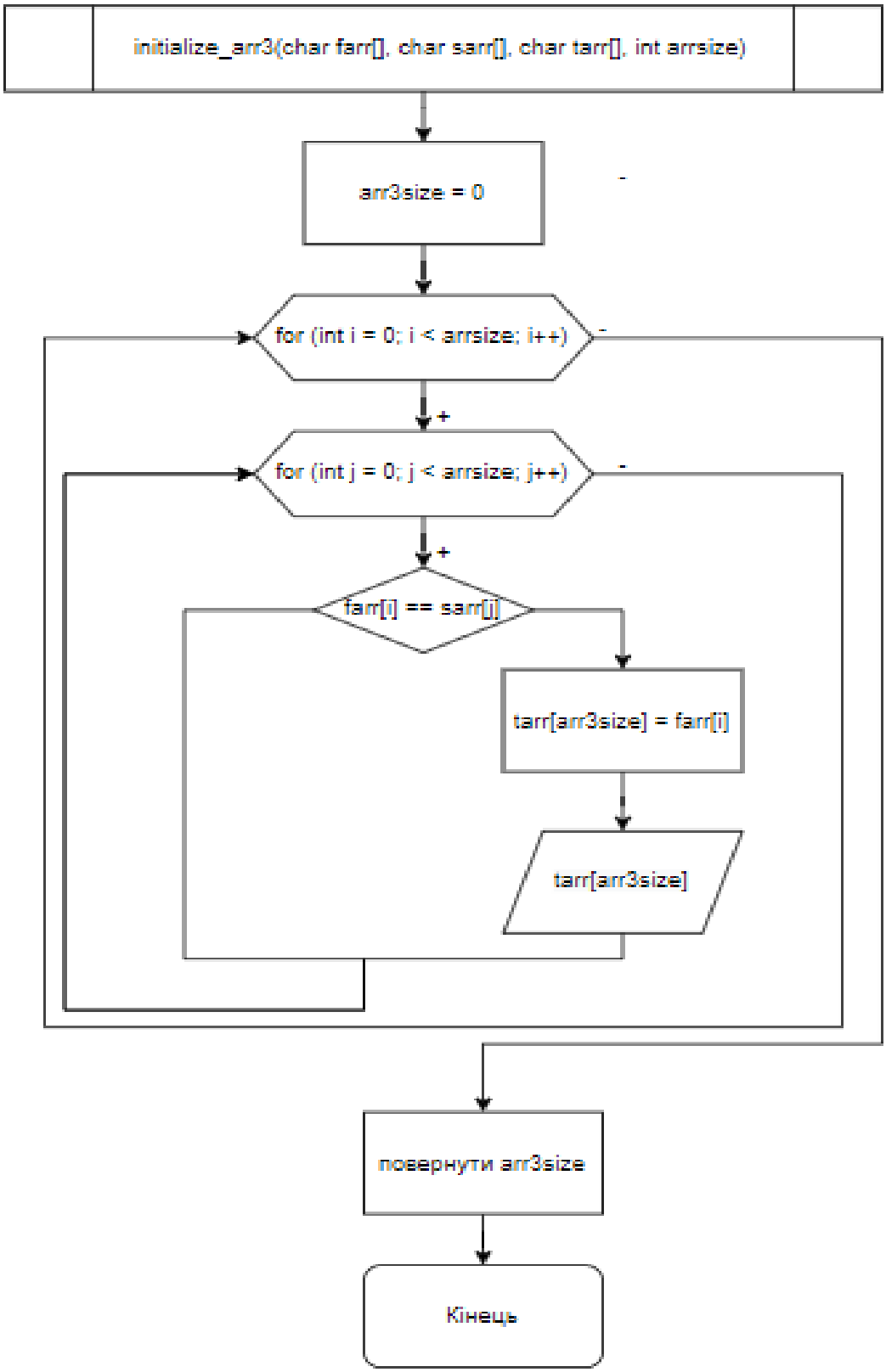
1.



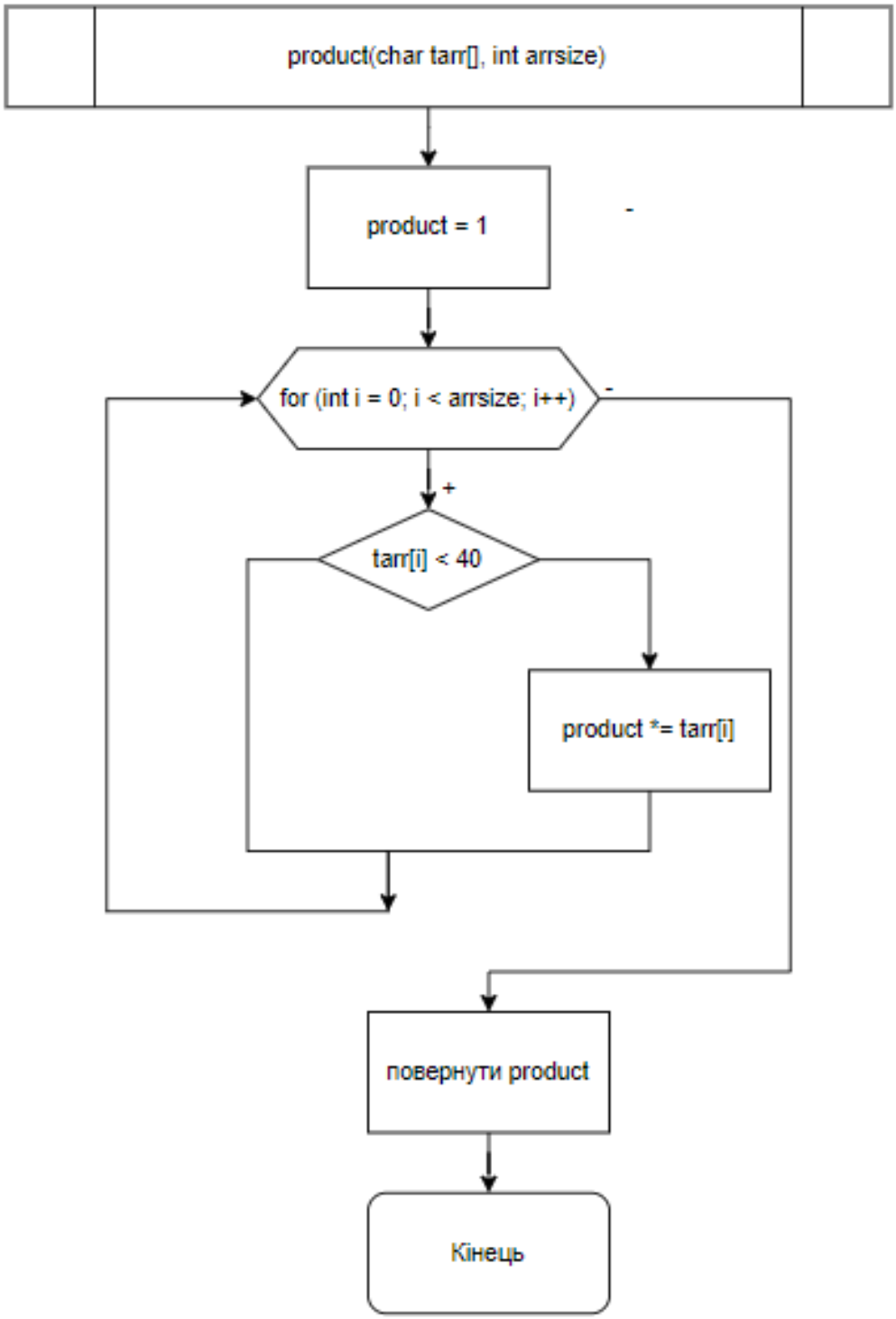
2.



3.


initialize_arr3(char farr[], char sarr[], char tarr[], int arrsize)
arr3size = 0
for (int i = 0; i < arrsize; i++)
for (int j = 0; j < arrsize; j++)
farr[i] == sarr[j]
tarr[arr3size] = farr[i]
tarr[arr3size]
повернути arr3size
Кінець

4.


product(char tarr[], int arrsize)
product = 1
for (int i = 0; i < arrsize; i++)
tarr[i] < 40
product *= tarr[i]
повернути product
Кінець

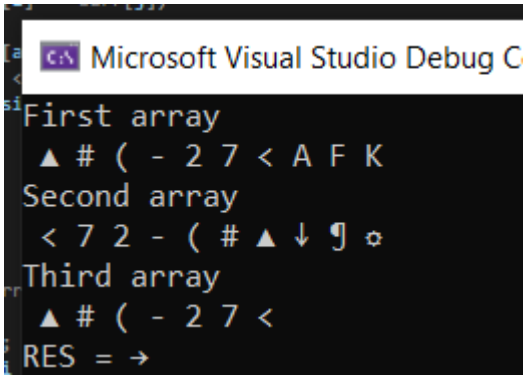
5) **Код та результат**

trying.cpp

trying (Global Scope)

```
28      int arr3size = 0;
29      for (int i = 0; i < arrsize; i++)
30      {
31          for (int j = 0; j < arrsize; j++)
32          {
33              if (farr[i] == sarr[j])
34              {
35                  tarr[arr3size] = farr[i];
36                  cout << " " << tarr[arr3size];
37                  arr3size++;
38              }
39          }
40      }
41      cout << endl;
42      return arr3size;
43  }
44  char product(char tarr[], int arrsize)
45  {
46      char product = 1;
47      for (int i = 0; i < arrsize; i++)
48      {
49          if (tarr[i] < 40)
50          {
51              product *= tarr[i];
52          }
53      }
54      return product;
55  }
56  int main()
57  {
58      int const arr12size = 10;
59      int arr3size;
60      char arr1[arr12size], arr2[arr12size], arr3[arr12size];
61      cout << "First array" << endl;
62      initialize_arr1(arr1, arr12size);
63
64      cout << "Second array" << endl;
65      initialize_arr2(arr2, arr12size);
66
67      cout << "Third array" << endl;
68      arr3size = initialize_arr3(arr1, arr2, arr3, arr12size);
69
70      char res = product(arr3, arr3size);
71
72      cout << "RES = " << res;
73  }
74
```

62 % No issues found



6) Випробування алгоритму:

Блок	Дія
	Початок
1	▲, #, (, -, 2, 7, <, A, F, K
2	<, 7, 2, -, (, #,▲, ↓, ¶, *
3	▲,#,(,-,2,7,<
4	Вивести res = ->
	Кінець

7)Висновок: дослідив методи послідовного пошуку у впорядкованих і невпорядкованих послідовностях та набув практичних навичок їх використання під час складання програмних специфікацій.