

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 27

Виконав студент ІП-15, Пономаренко Маргарита Альбертівна
(шифр, прізвище, ім'я, по батькові)

Перевірів _____
(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 набути практичних навичок їх використання під час складання програмних специфікацій.

Індивідуальне завдання

Варіант 27

Завдання

Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису трьох змінних індексованого типу з 10 символічних 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.

27	$95 + i$	$105 - i$	Суму елементів, коди яких більше 101
----	----------	-----------	--------------------------------------

Постановка задачі

Згенерувати два масиви за заданими формулами: $a = 95 + i$, $b = 105 - i$. Створити третій масив як перетин перших двох, а потім визначити коди яких елементів більше за 101 та замінити їх сумою елементи третього масиву. Результатом розв'язку є виведення початкових масивів, а також зміненого третього масиву.

Побудова математичної моделі

Генерувати перші два масиви “a” та “b” будемо за допомогою підпрограми `get_lists()`. Третій масив “c” генеруємо за допомогою підпрограми `get_c()`. У

третьому масиві буде перевірятися рівність елементів масивів “a” та “b” і у разі виконання умови спільний елемент буде додаватися у масив “c”. За допомогою підпрограми `get_code_sum()` знайдемо суму елементів коди яких більше 101.

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	<code>get_lists()</code>	Проміжні дані
Підпрограма, що генерує масив c	Процедура	<code>get_c()</code>	Проміжні дані
Підпрограма, що рахує суму елементів масиву c, код яких більший за 101	Дійсний	<code>get_code_sum()</code>	Результат
Лічильник масиву a	Цілочисельний	i	Проміжні дані
Лічильник масиву b	Цілочисельний	j	Проміжні дані
Виведення кінцевого масива	Процедура	<code>outp_c()</code>	Проміжні дані

В програмі для виразу $(x = x + 1)$ будемо використовувати `x += 1`, для перевірки на рівність будуть використані логічні вирази `==`, `!=`, `>`, `<`.

Розв'язання

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

Крок 1. Визначимо основні дії

Крок 2. Деталізуємо дію генерації масивів a та b

Крок 3. Деталізуємо дію генерації масиву c

Крок 4. Деталізуємо дію знаходження суми елементів масиву c, в яких код більший за 101

Крок 5. Деталізуємо дію виведення всіх необхідних масивів за допомогою підпрограм

Псевдокод

Крок 1

початок

ініціалізація масивів a, b, c

деталізація дії генерації масивів a, b

деталізація дії генерації масиву c

деталізація дії знаходження суми елементів масиву c, в яких код більший за 101

виведення результатів

кінець

Крок 2

початок

ініціалізація масивів a, b, c

get_lists()

деталізація дії генерації масиву c

деталізація дії знаходження суми елементів масиву c, в яких
код більший за 101

виведення результатів

кінець

Крок 3

початок

ініціалізація масивів a, b, c

get_lists()

get_c()

деталізація дії знаходження суми елементів масиву c, в яких
код більший за 101

виведення результатів

кінець

Крок 4

початок

ініціалізація масивів a, b, c

get_lists()

get_c()

get_code_sum()

виведення результатів

кінець

Крок 5

початок

ініціалізація масивів a, b, c

get_lists()

get_c()

get_code_sum()

outp_c()

кінець

Підпрограма 1:

get_lists()

початок

повторити для $i = 0, i < 10, i++$

$a[i] = 95 + i$

$b[i] = 105 - i$

все повторити

кінець

Підпрограма 2:

get_c()

початок

len = 0

повторити для $i = 0, i < 10, i++$

повторити для $j = 0, j < 10, j++$

якщо $a[i] == b[j]$

$c[i] == a[i]$

$len += 1$

все якщо

все повторити

все повторити

кінець

Підпрограма 3:

`get_code_sum()`

початок

$sum = 0$

повторити для $i = 0, i < len, i++$

якщо $c[i] > 101$

$sum += c[i]$

все якщо

все повторити

кінець

Підпрограма 4:

`outp_c()`

початок

виведення результатів

повторити для i від 0 до len

виведення $lst[i]$

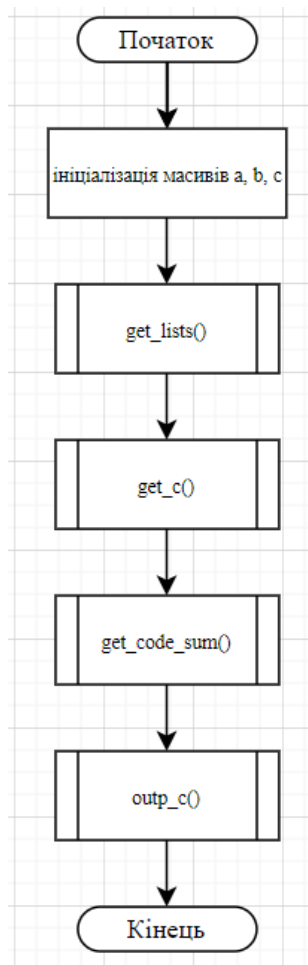
все повторити

кінець

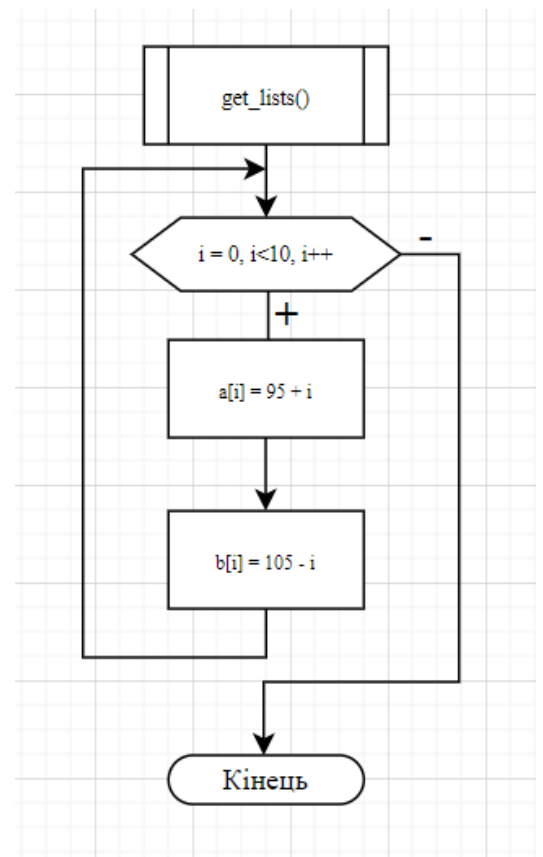
Блок схема

Основна програма:

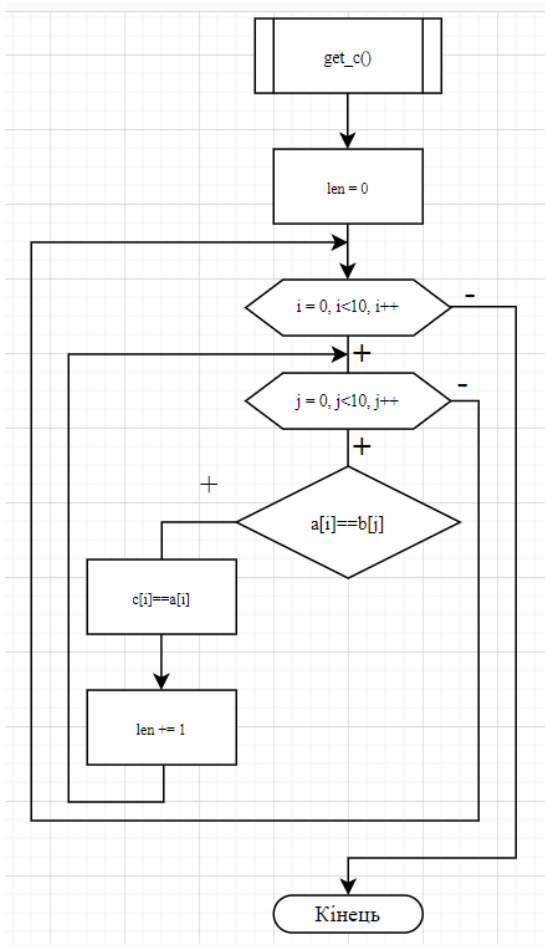
Крок 5



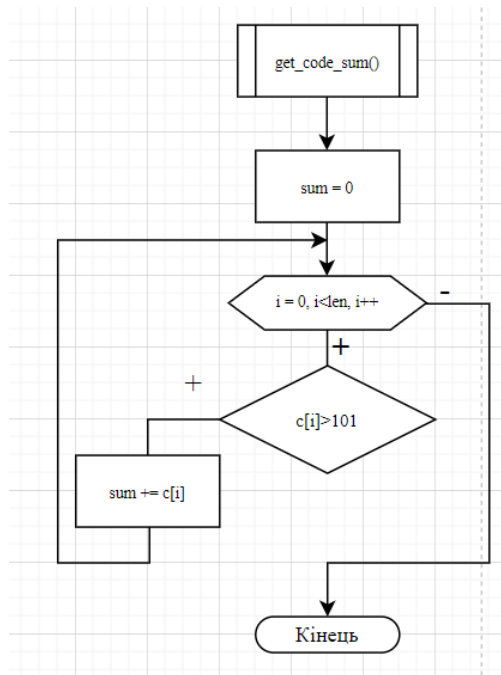
Підпрограма 1:



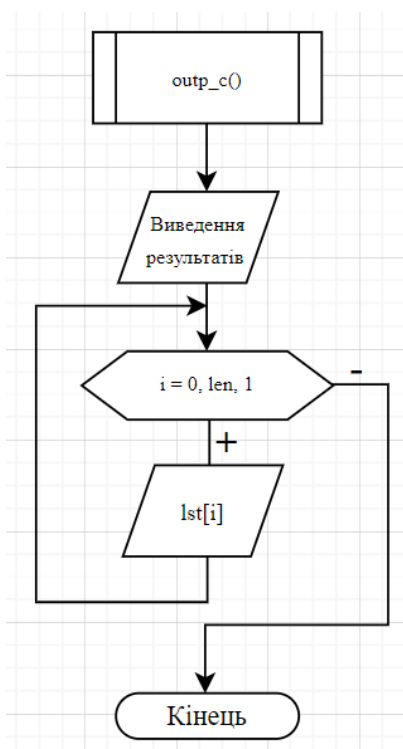
Підпрограма 2:



Підпрограма 3:



Підпрограма 4:



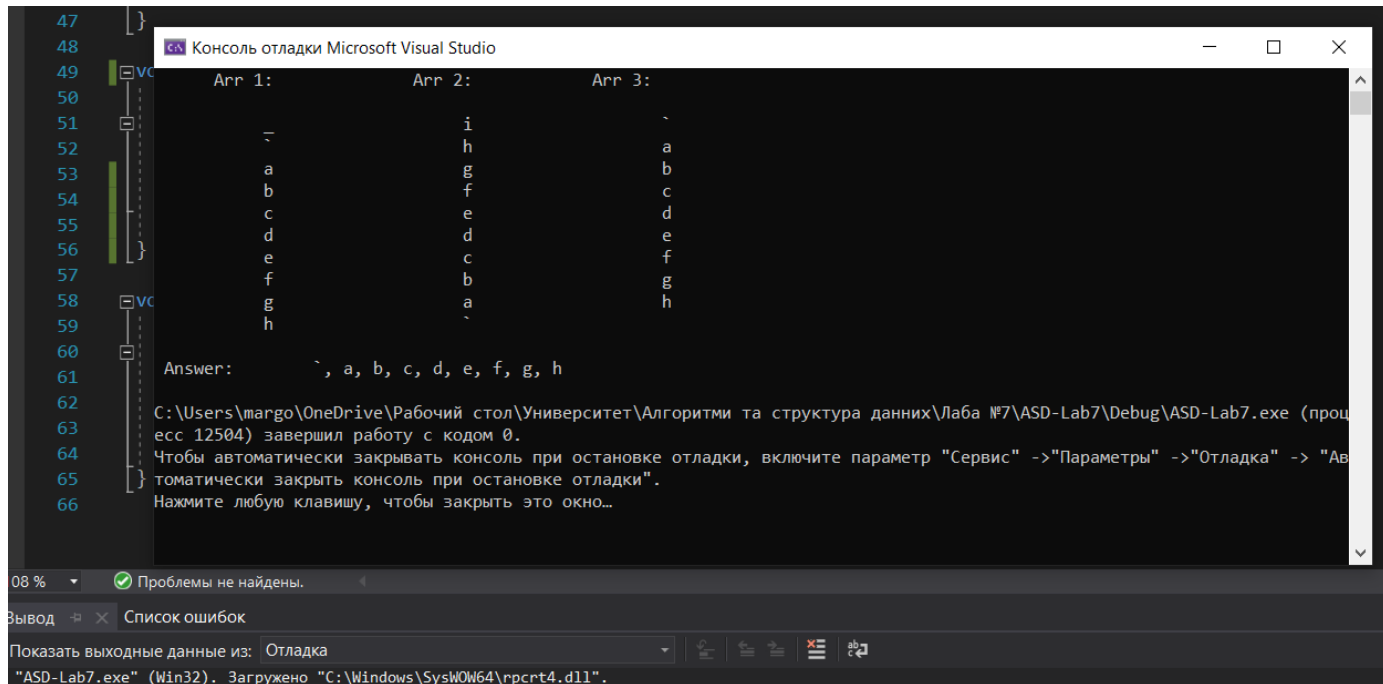
Код:

```

1  #include <stdio.h>
2  #include <iostream>
3  #include <iomanip>
4
5  using namespace std;
6
7  void outp_c(char lst[], int len);
8
9  int main() {
10     int code = 0, get_c(char[], char[], char[]);
11     char a[10], b[10], c[10], sum_el = 0;
12     void get_lists(char[], char[]), outp_abc(char[], char[], char[], int), get_code_sum(char[], int, char&, int&);
13     get_lists(a, b);
14     int len_c = get_c(a, b, c);
15     outp_abc(a, b, c, len_c);
16     get_code_sum(c, len_c, sum_el, code);
17     for (int i = 0; i < len_c; i++) if (c[i] == code) c[i] = sum_el;
18     outp_c(c, len_c);
19 }
20
21 void get_lists(char a[], char b[]) {
22     for (int i = 0; i < 10; i++) {
23         a[i] = 95 + i;
24         b[i] = 105 - i;
25     }
26 }
27
28 int get_c(char a[], char b[], char c[]) {
29     int len = 0;
30     for (int j = 0; j < 10; j++) {
31         for (int k = 0; k < 10; k++) {
32             if (a[j] == b[k]) {
33                 c[len] = a[j];
34                 len += 1;
35             }
36         }
37     }
38     return len;
39 }
40
41 void outp_abc(char a[], char b[], char c[], int len) {
42     cout << setw(12) << "Arr 1:" << setw(20) << "Arr 2:" << setw(20) << "Arr 3: \n" << endl;
43     for (int i = 0; i < len; i++) {
44         cout << setw(12) << a[i] << setw(20) << b[i] << setw(20) << c[i] << endl;
45     }
46     if (len < 10) cout << setw(12) << a[len] << setw(20) << b[len];
47 }
48
49 void get_code_sum(char c[], int len, char& sum_el, int& code) {
50     int sum = 0;
51     for (int i = 0; i < len; i++) {
52         if (int(c[i]) > 101) code = int(c[i]);
53         sum += int(c[i]);
54     }
55     sum_el = code;
56 }
57
58 void outp_c(char lst[], int len) {
59     cout << "\n\n Answer: \t";
60     for (int i = 0; i < len; i++) {
61         cout << lst[i];
62         if (i != len - 1) cout << ", ";
63         else cout << endl;
64     }
65 }

```

Тестування:



```
47 }
48
49 Arr 1:      Arr 2:      Arr 3:
50
51      ~          i          ~
52      a          h          a
53      b          g          b
54      c          f          c
55      d          e          d
56      e          d          e
57      f          c          f
58      g          b          g
59      h          a          h
60
61 Answer:      ~, a, b, c, d, e, f, g, h
62
63 C:\Users\margo\OneDrive\Рабочий стол\Университет\Алгоритми та структура даних\Лаба №7\ASD-Lab7\Debug\ASD-Lab7.exe (проц
64 есс 12504) завершил работу с кодом 0.
65 Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Ав
66 томатически закрывать консоль при остановке отладки".
    Нажмите любую клавишу, чтобы закрыть это окно...
```

Висновки

Ми дослідили методи послідовного пошуку у впорядкованих і невлпорядкованих послідовностях та набути практичних навичок їх використання під час складання програмних специфікацій. В наслідок виконання роботи ми створили алгоритм для знаходження суми елементів, коди яких більше 101. В процесі випробування ми розглянули єдиний можливий випадок і отримали результат $a = \{ _, \text{'}, a, b, c, d, e, f, g, h \}$ $b = \{ i, h, g, f, e, d, c, b, a, \text{'}$ $c = \{ _, a, b, c, d, e, f, g, h \}$.