

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 26

Виконав студент ПІ-11 Рябов Юрій Ігорович
(шифр, прізвище, ім'я, по батькові)

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

Індивідуальне завдання:

Варіант 26

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.

26	$2 * i + 42$	$54 - 2 * i$	Елементи, які менші за максимальний код
----	--------------	--------------	---

Постановка задачі

Необхідно описати 3 змінні індексованого типу з 10 символьних значень, присвоїти першим двом з них значення згідно з умовою, заповнити 3 змінну спільними їх елементами, знайти в ній елемент з максимальним кодом та знайти всі елементи 3 послідовності, що менше за нього. Вхідних даних достатньо, результатом виконання алгоритму є виведення всіх елементів 3 послідовності, що менше за його максимальний елемент.

Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Ім'я	Призначення
Перша послідовність	символьний	firstVector	Вхідне дане
Друга послідовність	символьний	secondVector	Вхідне дане
Третя послідовність	символьний	thirdVector	Проміжне дане
Максимальний код в 3 послідовності	цілий	maxCode	Проміжне дане, змінна підпрограми
Кількість елементів послідовності	цілий	size	Змінна підпрограм
Поточний елемент першої послідовності	символьний	currentFirst	Змінна підпрограми
Поточний елемент другої послідовності	символьний	currentSecond	Змінна підпрграми
Індекс першого порожнього елемента 3 послідовності	цілий	k	Змінна підпрограми

За допомогою підпрограми FillVectors з використанням арифметичної форми оператора повторення заповнимо перші два масиви, за допомогою підпрограми OutputVector з використанням арифметичної форми оператора повторення виведемо їх, за допомогою підпрограми FillWithCommon з використанням лінійного пошуку з використанням арифметичної форми оператора повторення та умовної форми оператора вибору заповнимо 3 масив, за допомогою підпрограми OutputVector виведемо його, за допомогою підпрограми FindMaxCode знайдемо найбільший елемент 3 масиву, за допомогою підпрограми OutputLessThan виведемо усі елементи 3 масиву крім найбільшого.

Розв'язання

Програмні специфікації запишемо у псевдокодi та графічній формi у вигляді блок-схеми.

Крок 1. Визначимо основні дії

Крок 2. Деталізуємо заповнення перших двох послідовностей

Крок 3. Деталізуємо виведення послідовності

Крок 4. Деталізуємо заповнення третьої послідовності

Крок 5. Деталізуємо знаходження максимального елемента послідовності

Крок 6. Деталізуємо виведення елементів послідовності, менших за певне значення

Псевдокод

Основна програма

Start

firstVector[10], secondVector[10], thirdVector[10]

FillVectors(firstVector, secondVector)

output "First array: "

OutputVector(firstVector)

output "Second array: "

OutputVector(secondVector)

FillWithCommon(firstVector, secondVector, thirdVector)

```

output "Third array: "
OutputVector(thirdVector)
maxCode = FindMaxCode(thirdVector)
output "Result: "
OutputLessThan(thirdVector, maxCode)

```

End

Підпрограми

FillVectors(vect1, vect2)

```

repeat for i from 1 to 11
    vect1[i-1] = 2*i + 42
    vect2[i-1] = 54 - 2*i
end repeat

```

End FillVectors

OutputVector(vect)

```

size = size(vect)
repeat for i from 0 to size
    output vect[i], " "
end repeat
output "\n"

```

End OutputVector

FillWithCommon(vect1, vect2, vect3)

```

k = 0
repeat for i from 0 to 10
    currentFirst = vect1[i]
    repeat for j from 0 to 10
        currentSecond = vect2[j]
        if currentSecond == currentFirst
            vect3[k] = currentFirst
            k += 1

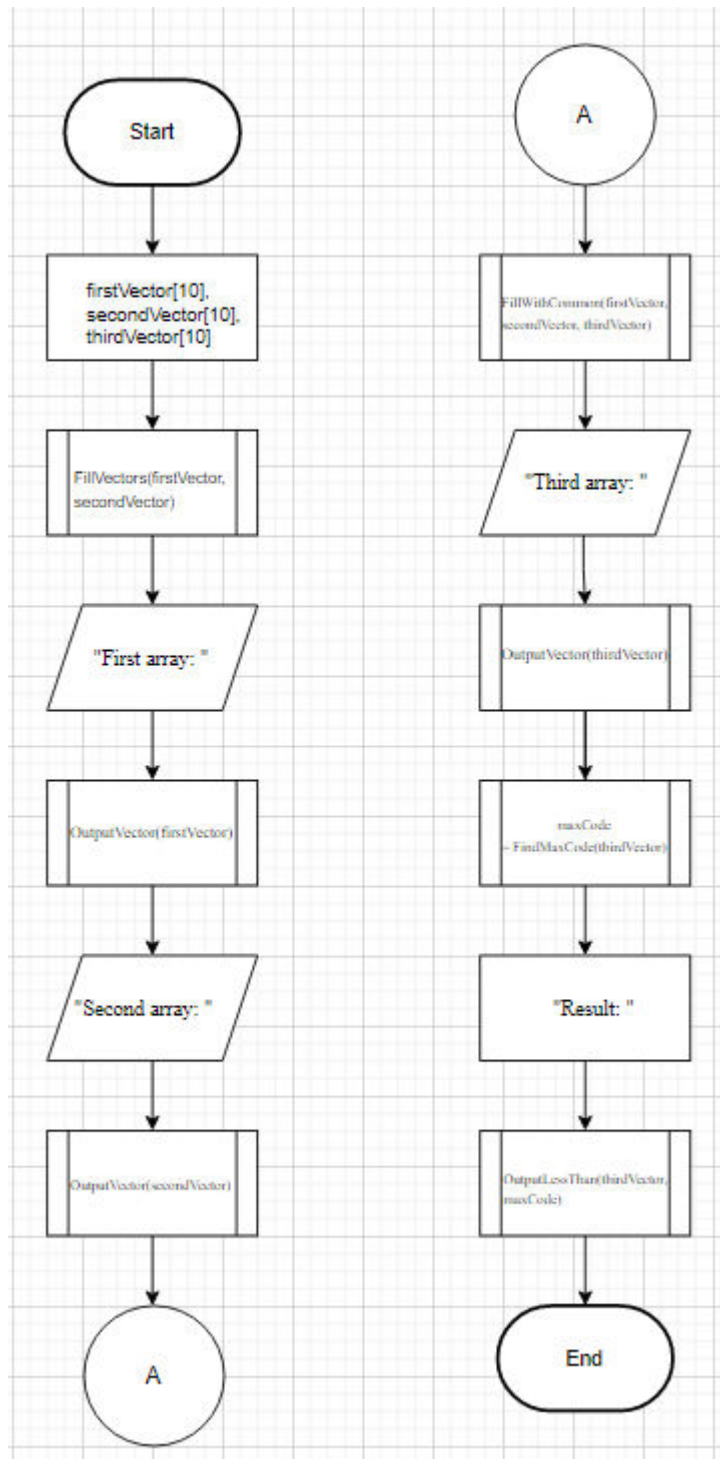
```

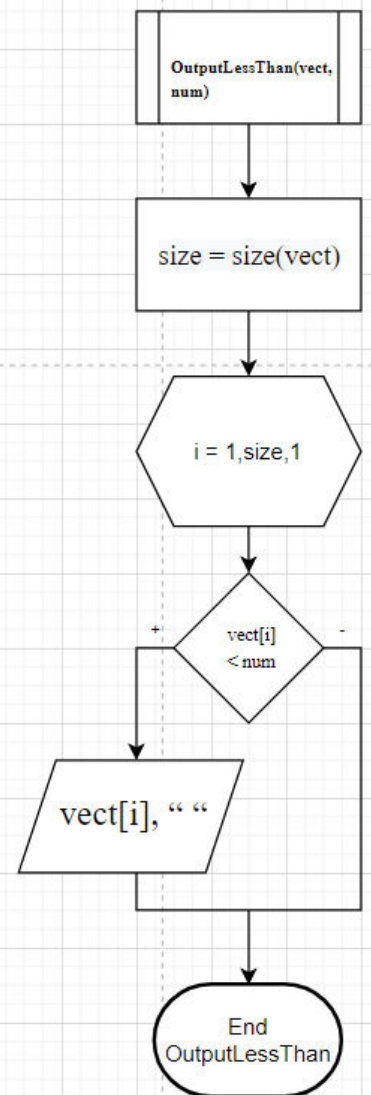
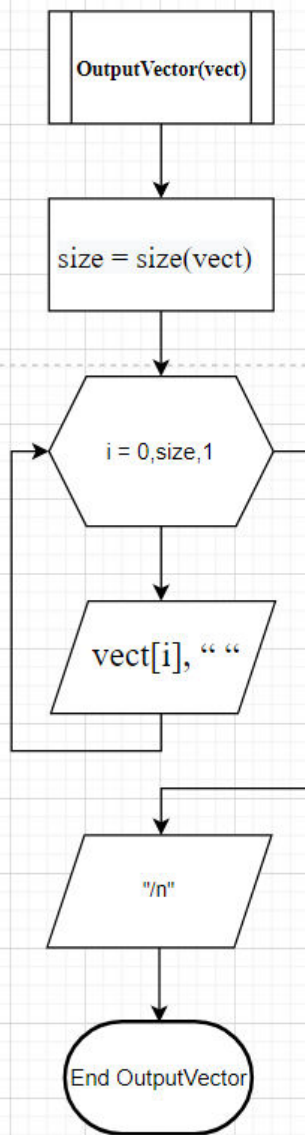
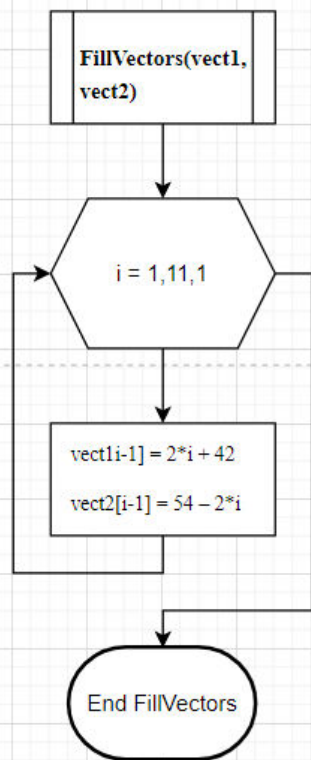
```
        end if
    end repeat
end repeat
End FillWithCommon
```

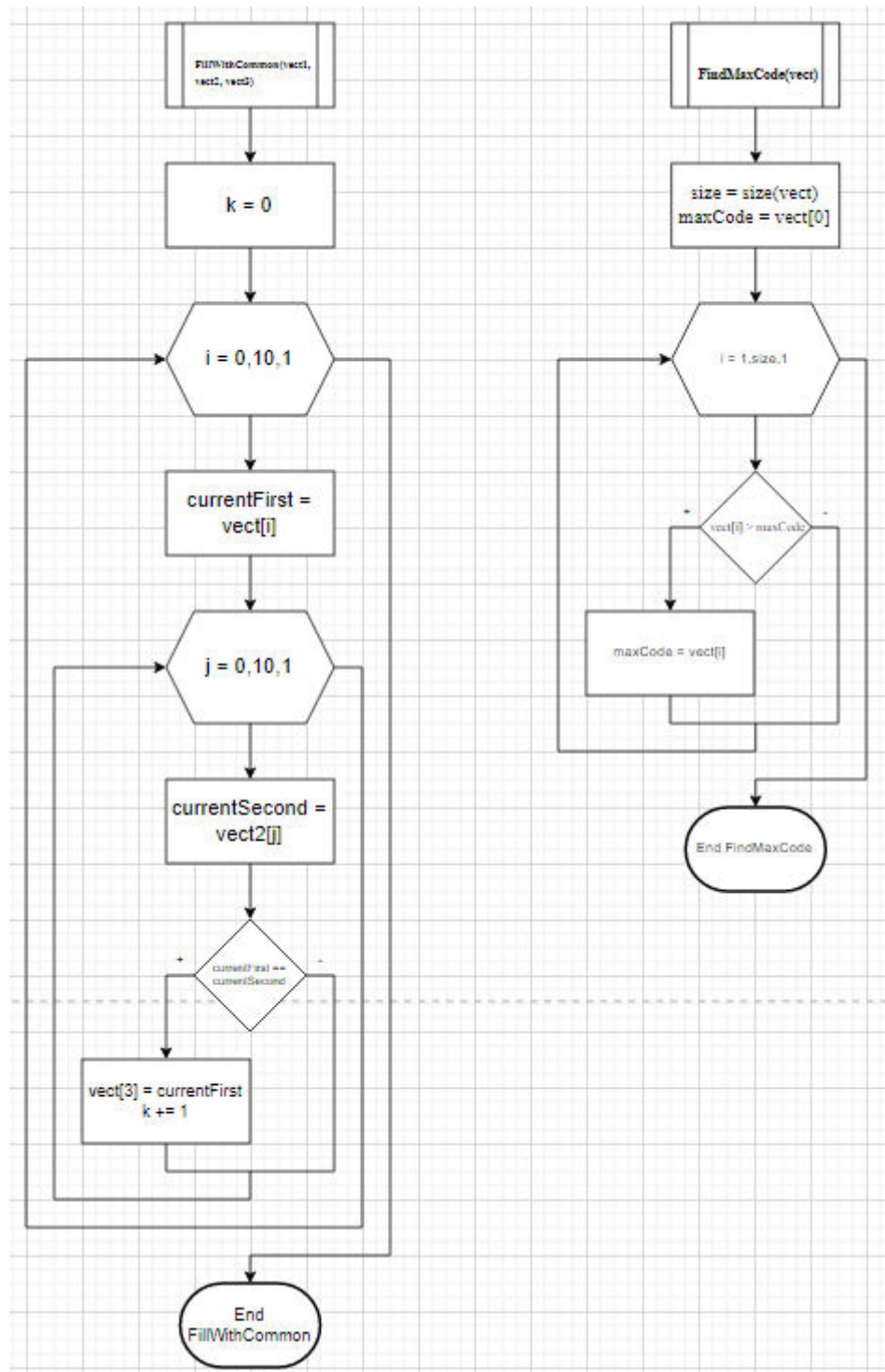
```
FindMaxCode(vect)
    maxCode = vect[1]
    size = size(vect)
    repeat for i from 0 to size
        if vect[i] > maxCode
            maxCode = vect[i]
        end if
    end repeat
    return maxCode
End FindMaxCode
```

```
OutputLessThan(vect, num)
    size = size(vect)
    repeat for i from 0 to size
        if vect[i] < num
            output vect[i], “ “
        end if
    end repeat
End OutputLessThan
```

Блок-схема







Програма на мові C++

```
#include <iostream>
#include <vector>

void FillVectors(std::vector<char>& vect1, std::vector<char>& vect2);
void OutputVector(const std::vector<char>& vect);
void FillWithCommon(const std::vector<char>& vect1, const std::vector<char>& vect2, std::vector<char>& vect3);
int FindMaxCode(const std::vector<char>& vect);
void OutputLessThan(const std::vector<char>& vect, int num);

int main()
{
    std::vector<char> firstVector(10), secondVector(10), thirdVector(10);
    int maxCode;

    FillVectors(firstVector, secondVector);

    std::cout << "First array: ";
    OutputVector(firstVector);
    std::cout << "Second array: ";
    OutputVector(secondVector);

    FillWithCommon(firstVector, secondVector, thirdVector);

    std::cout << "Third array: ";
    OutputVector(thirdVector);

    maxCode = FindMaxCode(thirdVector);

    std::cout << "Result: ";
    OutputLessThan(thirdVector, maxCode);

    system("pause");
}

void FillVectors(std::vector<char>& vect1, std::vector<char>& vect2)
{
    for (int i = 1; i < 11; i++)
    {
        vect1[i-1] = (char)(2 * i + 42);
        vect2[i-1] = (char)(54 - 2 * i);
    }
}

void OutputVector(const std::vector<char>& vect)
{
    int size = vect.size();
    for (int i = 0; i < size; i++)
    {
        std::cout << vect[i] << " ";
    }
    std::cout << "\n";
}

void FillWithCommon(const std::vector<char>& vect1, const std::vector<char>& vect2, std::vector<char>& vect3)
{
    char currentFirst, currentSecond;
    int k = 0;
    for (int i = 0; i < 10; i++)
    {
        currentFirst = vect1[i];
        for (int j = 0; j < 10; j++)
        {
            currentSecond = vect2[j];
            if (currentFirst == currentSecond)
            {
                vect3[k] = currentSecond;
                k++;
            }
        }
    }
}
```

```

    }
}

int FindMaxCode(const std::vector<char>& vect)
{
    int maxCode = (int)vect[0];
    int size = vect.size();

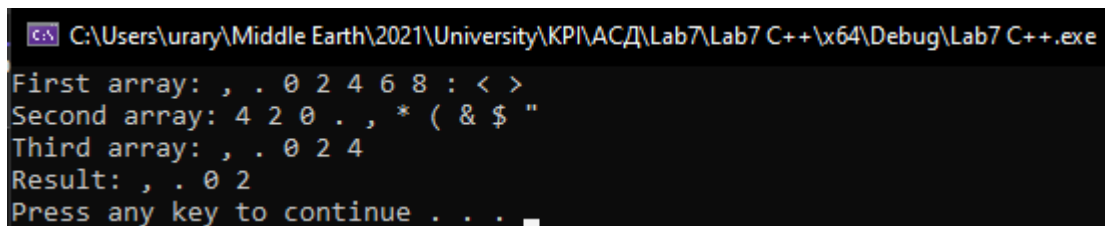
    for (int i = 1; i < size; i++)
    {
        if ((int)vect[i] > maxCode)
        {
            maxCode = (int)vect[i];
        }
    }

    return maxCode;
}

void OutputLessThan(const std::vector<char>& vect, int num)
{
    int size = vect.size();
    for (int i = 0; i < size; i++)
    {
        if ((int)vect[i] < num)
        {
            std::cout << vect[i] << " ";
        }
    }
    std::cout << "\n";
}

```

Виконання програми



```

C:\Users\urary\Middle Earth\2021\University\KPI\ACД\Lab7\Lab7 C++\x64\Debug\Lab7 C++.exe
First array: , . 0 2 4 6 8 : < >
Second array: 4 2 0 . , * ( & $ "
Third array: , . 0 2 4
Result: , . 0 2
Press any key to continue . . . _

```

Перевірка

Блок	Дія
	Початок
1	Виведення: First array: , . 0 2 4 6 8 : < > (символи з кодами з 44 по 62 включно з кроком 2)
2	Виведення: Second array: 4 2 0 . , * (& \$ " (символи з кодами з 52 по 34 включно з кроком 2)
3	Виведення: Third array: , . 0 2 4 (спільні елементи перших 2 послідовностей)
4	maxCode = 52 (код числа 4)
5	Виведення: Result: , . 0 2
	Кінець

Висновок

Отже, ми дослідили методи послідовного пошуку в послідовностях та набули

практичних навичок їх використання, створивши алгоритм з використанням підпрограми, в якій методом лінійного пошуку знаходили спільні елементи двох послідовностей.