

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 30

Виконав студент ІІ-12 Тарасюк Євгеній Сергійович

Перевірив _____

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.

Задача 30.

Завдання

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.

30	43 - i	37 + i	Добуток елементів, коди яких більше 40
----	--------	--------	--

Розв'язок.

1. Постановка задачі.

Програма працює без вхідних даних. Для ініціювання змінних та обчислень використовуватимемо арифметичні цикли. Результатом розв'язку є три списки символів та натуральне число. Використовуватимемо стандартні логічні та арифметичні операції, а також функції.

2. Побудова математичної моделі

Таблиця змінних та функцій:

Змінні та функції	Тип	Ім'я	Призначення
Масив А	Масив символічних змінних	listA	Збереження символів у першому масиві
Масив В	Масив символічних змінних	listB	Збереження символів у другому масиві
Масив С	Масив символічних змінних	listC	Збереження символів у третьому масиві
Розмір С	Натуральне число	cSize	Розмір масива С
Добуток	Натуральне число	res	Добуток елементів масива С
i	Ціле число	i	Допоміжна змінна для роботи арифметичних циклів
Заповнення масивів	Функція	fillLists (listA, listB)	Заповнює масиви А та В згідно з умовою задачі
Аргументи функції мають такі ж ролі, що й змінні, згадані вище			
Кількість однакових символів	Функція, що повертає натуральне число	equalCount (listA, listB)	Знаходить кількість однакових символів у масивах А та В
Аргументи функції мають такі ж ролі, що й змінні, згадані вище			
кількість	Ціле число	count	Змінна для рахунку однакових символів
j	Ціле число	j	Допоміжна змінна для роботи арифметичних циклів

Заповнення C	Функція	fillC(listA, listB, listC)	Заповнює масив C символами, що є в обох масивах A та B
Аргументи функції мають такі ж ролі, що й змінні, згадані вище			
j	Ціле число	j	Допоміжна змінна для роботи арифметичних циклів
k	Ціле число	k	Допоміжна змінна для роботи арифметичних циклів
Множення	Функція, що повертає натуральне число	mult(list, listSize)	Множить між собою чисельні коди символів у масиві
Масив	Масив символьних змінних	list	Масив-аргумент
Розмір масиву	Натуральне число	listSize	Розмір масиву-аргументу
Добуток	Натуральне число	res	Добуток елементів
Виведення масиву	Функція	print(list, listSize)	Виводить у рядок усі символи масиву (використовується в коді програми, у псевдокоді та блок-схемі не згадується)
Аргументи функції мають такі ж ролі, що й змінні, згадані вище			

3. Псевдокод алгоритму

Початок

```
listA = [10]
listB = [10]
fillLists(listA, listB)
cSize = equalCount(listA, listB)
listC = [cSize]
fillC(listA, listB, listC)
res = mult(listC, cSize)
Виведення listA, listB, listC, res
```

Кінець.

Функція fillLists(listA, listB)

Початок

```
Повторити для i на проміжку [0; 10)
    listA[i] = 37 + i
    listB[i] = 43 - i
```

Все повторити

Повернення

Кінець

Функція equalCount(listA, listB)

Початок

```
count = 0
Повторити для i на проміжку [0; 10)
    Повторити для j на проміжку [0; 10)
        Якщо listA[i] == listB[j]
            count += 1
    Все якщо
Все повторити
Все повторити
Повернення count
```

Кінець

Функція fillC(listA, listB, listC)

Початок

i = 0

j = 0

k = 0

Повторити поки i < 10

Повторити поки j < 10

Якщо listA[i] == listB[j]

listC[k] = listA[i]

k += 1

i += 1

j = 0

Інакше

j += 1

Все якщо

Все повторити

i += 1

j = 0

Все повторити

Повернення

Кінець

Функція mult(listC, cSize)

Початок

res = 1

Повторити для i на проміжку [0; listSize)

Якщо list[i] > 40

res *= list[i]

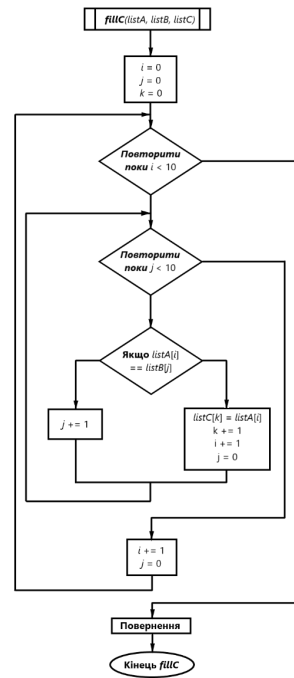
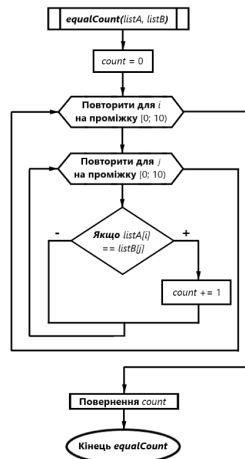
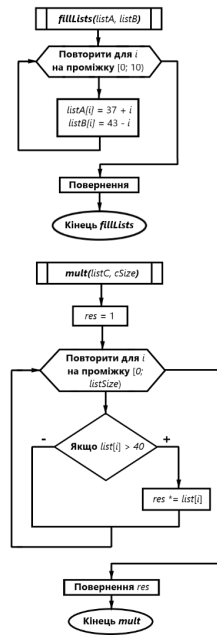
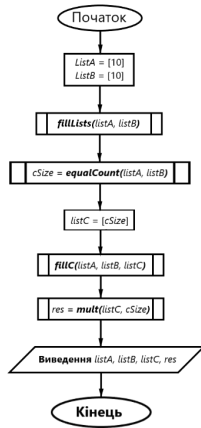
Все якщо

Все повторити

Повернення res

Кінець

4. Блок-схема алгоритму



5. Випробування алгоритму.

Перевіримо правильність алгоритму для різних вхідних даних:

Крок алгоритму	Тест
$listA = [10]$ $listB = [10]$	$listA = \{ , , , , , , , \}$, $listB = \{ , , , , , , , \}$
$fillLists(listA, listB)$	$listA = \{ , , , , , , , \}$, $listB = \{ , , , , , , , \}$
$i = 0$	$listA = \{ \%, , , , , , , \}$, $listB = \{ +, , , , , , , \}$
$i = 1$	$listA = \{ \%, \&, , , , , , , \}$, $listB = \{ +, *, , , , , , , \}$
...	...
$i = 9$	$listA = \{ \%, \&, ', (,), *, +, ,, -, . \}$, $listB = \{ +, *,), (, ', \&, \%, \$, \#, \{ \}$
$cSize = equalCount(listA, listB)$	
$i = 0, j = 0$	$listA[i] = \%$ $listB[j] = +$ $count = 0$
...	...
$i = 0, j = 6$	$listA[i] = \%$ $listB[j] = \%$ $count = 1$
...	...
$i = 5, j = 9$	$listA[i] = *$ $listB[j] = \{$ $count = 6$
$i = 6, j = 0$	$listA[i] = +$ $listB[j] = +$ $count = 7$

...	...
i = 9, j = 9	listA[i] = '.' listB[j] = "" count = 7
listC = [cSize] fillC(listA, listB, listC)	listC = { , , , , , } listA = {%, &, `, (,), *, +, ,, -, .}, listB = {+, *,), (, `, &, %, \$, #, "{
i = 0, j = 0	listA[i] = '%' listB[j] = '+' listC = { , , , , , }
...	
i = 0, j = 6	listA[i] = '%' listB[j] = '%' listC = {%, , , , , }
...	
i = 5, j = 9	listA[i] = '*' listB[j] = "" listC = {%, &, `, (,), *, }
i = 6, j = 0	listA[i] = '+' listB[j] = '+' listC = {%, &, `, (,), *, +}
...	
i = 9, j = 9	listA[i] = '.' listB[j] = "" listC = {%, &, `, (,), *, +}
res = mult(listC, cSize)	listC = {%, &, `, (,), *, +}, cSize = 7
i = 0	listC[i] = % (37) res = 1
i = 1	listC[i] = & (38) res = 1

i = 2	listC[i] = ` (39) res = 1
i = 3	listC[i] = ((40) res = 1
i = 4	listC[i] =) (41) res = 41
i = 5	listC[i] = * (42) res = 1722
i = 6	listC[i] = + (43) res = 74046
Виведення listA, listB, listC, res	listA: % & ` () * + , - . listB: + *) (` & % \$ # “ listC: % & ` () * + res: 74046

6. Код программы (C++)

```
#include <iostream>
#include <iomanip>
using namespace std;

void fillLists(char[], char[]);
int equalCount(char[], char[]);
void fillC(char[], char[], char[]);
int mult(char[], int);
void print(char[], int);

int main()
{
    char listA[10];
    char listB[10];
    fillLists(listA, listB);
    int cSize = equalCount(listA, listB);
    char *listC = new char[cSize];
    fillC(listA, listB, listC);
    cout << "List 1:\n";
    print(listA, 10);
    cout << "List 2:\n";
    print(listB, 10);
    cout << "List 3:\n";
    print(listC, cSize);
    cout << mult(listC, cSize) << endl;
    delete [] listC;
}

void fillLists(char listA[], char listB[])
{
    for (int i = 0; i < 10; i++)
    {
        listA[i] = 37 + i;
        listB[i] = 43 - i;
    }
}

int equalCount(char listA[], char listB[])
{
    int count = 0;
    for (int i = 0; i < 10; i++)
    {
        for (int j = 0; j < 10; j++)
        {
            if (listA[i] == listB[j])
            {
                count += 1;
            }
        }
    }
    return count;
}

void fillC(char listA[], char listB[], char listC[])
{

```

```

int i = 0;
int j = 0;
int k = 0;
while (i < 10)
{
    while (j < 10)
    {
        if (listA[i] == listB[j])
        {
            listC[k] = listA[i];
            k += 1;
            i += 1;
            j = 0;
        }
        else
        {
            j += 1;
        }
    }
    i += 1;
    j = 0;
}

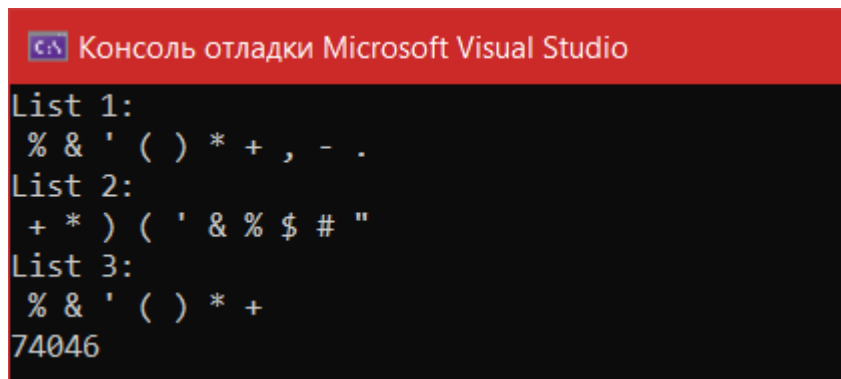
int mult(char list[], int listSize)
{
    int res = 1;
    for (int i = 0; i < listSize; i++)
    {
        if (list[i] > 40)
        {
            res *= list[i];
        }
    }
    return res;
}

void print(char list[], int listSize)
{
    for (int i = 0; i < listSize; i++)
    {
        cout << setw(2) << list[i];
    }
    cout << endl;
}

```

7. Випробування коду

Результати випробування програми



```
Консоль отладки Microsoft Visual Studio
List 1:
% & ' ( ) * + , - .
List 2:
+ * ) ( ' & % $ # "
List 3:
% & ' ( ) * +
74046
```

8. Висновки

Було дослідити методи послідовного пошуку у невідсортованій послідовності символів та набуто практичних навичок їх використання під час складання програми для знаходження однакових символів та символів з особливими значеннями.