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 арифметичних циклічних алгоритмів» Варіант 17

Виконав студент ІП-15 Куркчі Юрій
(шифр, прізвище, ім'я, по батькові)

Перевірив
(прізвище, ім'я, по батькові)

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 17

№	Вираз для обчислення елемента		Знайти
	1-го масиву	2-го масиву	
17	5 * i + 25	55 - 5 * i	Добуток елементів, коди яких менше 82

Постановка задачі

Заданий алгоритм повинен:

1. Створити три змінні індексованого типу з 10 символьних значенньь.
2. Ініціювати перші 2 змінні за формулами $5i + 25$ та $55 - 5i$, де $i \in [1,10]$.
3. Ініціювати трерю змінну спільними значеннями двох попередніх.
4. Знайти добуток елементів з третьої змінної, коди яких менше 82.

Побудова математичної моделі

Таблиця змінних

Змінна	Тип	Ім’я	Призначення
Розмірність масивів	Натуральний	size	Початкові дані
Перший масив	Символьний	F	Проміжні дані
Другий масив	Символьний	S	Проміжні дані
Третій масив	Символьний	thirdArray	Проміжні дані
Додатковий масив	Символьний	newArray	Проміжні дані

Добуток кодів	Натуральний	product	Проміжні дані
Рівність елементів	Логічний	match	Проміжні дані
Результат добутку	Логічний	result	Кінцеві дані

Використані функції

- char(x) – повертає символьне значення числа
- int(x) – повертає числове значення символу
- a%b – повертає остачу від ділення числа a на число b.

Власні функції

- generateFirstArray(ціле число) – повертає перший символьний масив
- generateFirstArray(ціле число) – повертає другий символьний масив
- generateFirstArray(одновимірний символьний масив, одновимірний символьний масив, ціле число) – повертає третій символьний масив згенерований на основі перших двух.
- product(одновимірний символьний масив, ціле число) – повертає добуток елементів, коди яких менше 82.

1. Згенеруємо перші 2 масиви за формулами $5i + 25$ та $55 - 5i$.
2. Згенеруємо третій масив шляхом перевірки кожної пари елементів на рівність та їх внесення до масиву. Інші елементи заповнимо нулями.
3. Розрахуємо добуток ненульових елементів третього масиву.

Розв’язання

1. Визначимо основні дії.
2. Деталізуємо дію визначення першого масиву.
3. Деталізуємо дію визначення другого масиву.
4. Деталізуємо дію визначення третього масиву.
5. Деталізуємо дію обрахунку добутків ненульових елементів третього масиву.
6. Деталізуємо дію визначення елементів першого масиву за допомогою ітераційної форми повторення.
7. Деталізуємо дію визначення елементів другого масиву за допомогою ітераційної форми повторення.
8. Деталізуємо дію визначення елементів третього масиву за допомогою ітераційної форми повторення, умовної та альтернативної форм вибору.

9. Деталізуємо дію обрахунку добутків елементів третього масиву за допомогою ітераційної форми повторення.

Псевдокод алгоритму

Крок 1 **Початок**
    size:=10
    Визначення першого масиву
Визначення другого масиву
    Визначення третього масиву
    Обрахунок добутків елементів третього масиву
    Виведення result
Кінець

Підпрограма
    generateFirstArray(size**)**
newArray[size]
    Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
    generateSecondArray(size**)**
newArray[size]
    Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
    generateThirdArray(firstArray, secondArray, size**)**
newArray[size]
    Визначення елементів третього масиву
повернути newArray **Все підпрограма**

Підпрограма
product(thirdArray, size**)**
product:=1
Обрахунок добутку елементів
повернути char(product) **Все підпрограма**

Крок 2
Початок
size:=10
firstArray[size]=generateFirstArray(size)
Визначення другого масиву
Визначення третього масиву
Обрахунок добутків елементів третього масиву
Виведення result
Кінець

Підпрограма
generateFirstArray(size**)**
newArray[size]
Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
generateSecondArray(size**)**
newArray[size]
Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
generateThirdArray(firstArray, secondArray, size**)**
newArray[size]
Визначення елементів третього масиву
повернути newArray **Все підпрограма**

Підпрограма
product(thirdArray, size**)**
product:=1

Обрахунок добутку елементів
повернути char(product) **Все**
підпрограма

size:=10
firstArray[size]=generateFirstArray(size)
secondArray[size]=generateSecondArray(size)

Крок 3

Початок

Визначення третього масиву
Обрахунок добутків елементів третього масиву
Виведення result

Кінець

Підпрограма
generateFirstArray(size**)**
newArray[size]
Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
generateSecondArray(size**)**
newArray[size]
Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
generateThirdArray(firstArray, secondArray, size**)**
newArray[size]
Визначення елементів третього масиву
повернути newArray **Все підпрограма**

Підпрограма
product(thirdArray, size**)**
product:=1 обрахунок добутку елементів **повернути** char(product) **Все підпрограма**
thirdArray[size]=generateThirdArray(size) Обрахунок добутків елементів третього масиву

Крок 4

Початок
size:=10
firstArray[size]=generateFirstArray(size)
secondArray[size]=generateSecondArray(size)
Виведення result
Кінець

Підпрограма
generateFirstArray(size**)**
newArray[size]
Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
generateSecondArray(size**)**
newArray[size]
Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
generateThirdArray(firstArray, secondArray, size**)**
newArray[size]
Визначення елементів третього масиву
повернути newArray **Все підпрограма**

Підпрограма
product(thirdArray, size**)**
product:=1 обрахунок добутку елементів **повернути** char(product) **Все підпрограма**
thirdArray[size]=generateThirdArray(size)
result=product(thirdArray, size)
Виведення result
Кінець

Підпрограма
generateFirstArray(size**)**
newArray[size]

Крок 5

Початок
size:=10
firstArray[size]=generateFirstArray(size)
secondArray[size]=generateSecondArray(size)
Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
generateSecondArray(size**)**
newArray[size]
Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
generateThirdArray(firstArray, secondArray, size**)**
newArray[size]
Визначення елементів третього масиву
повернути newArray **Все підпрограма**

Підпрограма
product(thirdArray, size**)**
product:=1
Обрахунок добутку елементів
повернути char(product) **Все підпрограма**
thirdArray[size]=generateThirdArray(size)
result=product(thirdArray, size)
Виведення result
Кінець

Підпрограма
generateFirstArray(size**)**
newArray[size] **повторити для** i **від** 1 **до** size
newArray[i]:=char(25+5*i)
повернути newArray **Все підпрограма**

Підпрограма

Крок 6

Початок
size:=10
firstArray[size]=generateFirstArray(size)
secondArray[size]=generateSecondArray(size)
generateSecondArray(size**)**
newArray[size]
Визначення елементів першого масиву
повернути newArray **Все підпрограма**

Підпрограма
generateThirdArray(firstArray, secondArray, size**)**
newArray[size]
Визначення елементів третього масиву
повернути newArray **Все підпрограма**

Підпрограма
product(thirdArray, size**)**
product:=1
Обрахунок добутку елементів
повернути char(product) **Все підпрограма**
thirdArray[size]=generateThirdArray(size)
result=product(thirdArray, size)
Виведення result
Кінець

Підпрограма
generateFirstArray(size**)**
newArray[size] **повторити**
для i **від** 1 **до** size
newArray[i]:=char(25+5*i)
все повторити **повернути**
newArray **Все підпрограма**

Підпрограма
generateSecondArray(size**)**
newArray[size]
повторити **для** i **від** 1 **до**
size newArray[i]:=char(55-

Крок 7

Початок
size:=10
firstArray[size]=generateFirstArray(size)
secondArray[size]=generateSecondArray(size)
5*i) **все повторити**
повернути newArray **Все**
підпрограма

Підпрограма
generateThirdArray(firstArray, secondArray, size**)**
newArray[size]
Визначення елементів третього масиву
повернути newArray **Все підпрограма**

Підпрограма
product(thirdArray, size**)**
newArray[size]
Обрахунок добутку елементів
повернути char(product)
Все підпрограма
thirdArray[size]=generateThirdArra
y(size)
result=product(thirdArray, size)
Виведення result
Кінець

Підпрограма
generateFirstArray(size**)**
newArray[size] **повторити**
для i **від** 1 **до** size
newArray[i]:=char(25+5*i)
все повторити **повернути**
newArray **Все підпрограма**

Підпрограма
generateSecondArray(size**)**
newArray[size] **повторити**
для i **від** 1 **до** size
newArray[i]:=char(55-5*i) **все**

Крок 8

Початок
size:=10
firstArray[size]=generateFirstArray(size)
secondArray[size]=generateSecondArray(size)
повторити **повернути**
newArray **Все підпрограма**

Підпрограма
generateThirdArray(firstArray, secondArray, size**)**
newArray[size]
повторити **для** i **від**
1 **до** size
match=false
повторити **для** j
від 1 **до** size
якщо firstArray[i]==secondArray[j]
то
match:=true **все**
якщо **все**
повторити **якщо**
match==true
то
newArray[i]:=firstArray[i]
інакше
newArray[i]:=char(0)
все якщо **все повторити**
все повторити **повернути**
newArray **Все підпрограма**

Підпрограма
product(thirdArray, size**)**
product:=1
Обрахунок добутку
елементів **повернути**
char(product) **Все підпрограма**

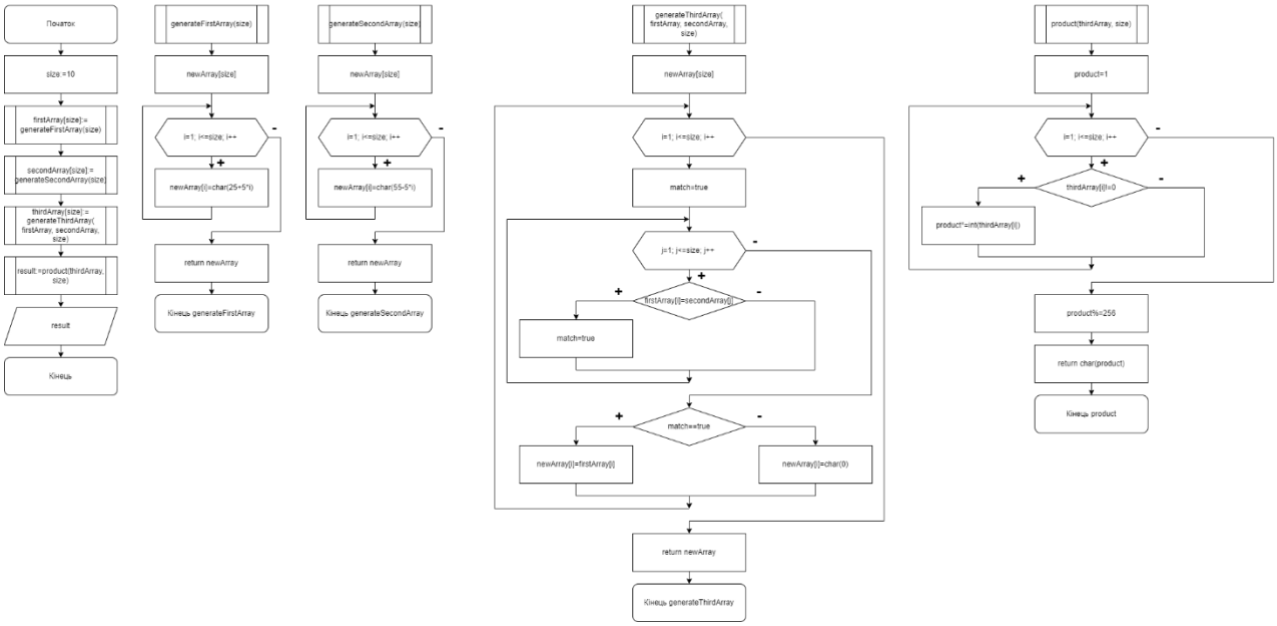
Крок 9

Початок
size:=10
firstArray[size]=generateFirstArray(size)
secondArray[size]=generateSecondArray(size)
thirdArray[size]=generateThirdArray(size)
result=product(thirdArray, size)
Виведення result
Кінець

Підпрограма
generateFirstArray(size**)**
newArray[size] **повторити**
для i **від** 1 **до** size
newArray[i]:=char(25+5*i)
все повторити **повернути**
newArray **Все підпрограма**

Підпрограма
generateSecondArray(thirdArray, size**)**
newArray[size]
повторити **для** i **від** 1 **до**
size
newArray[i]:=char(55-5*i)
все повторити **повернути**
newArray **Все підпрограма**

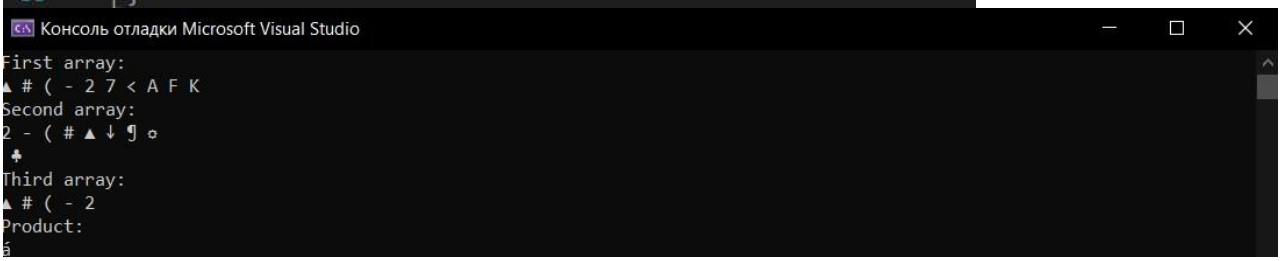
```
Підпрограма
    generateThirdArray(firstArray, secondArray, size)
    newArray[size]
повторити    для i від
1 до size
match=false
повторити        для j
від 1 до size
        якщо firstArray[i]==secondArray[j]
            то
match:=true            все
якщо        все
повторити        якщо
match==true
        то
            newArray[i]:=firstArray[i]
інакше
            newArray[i]:=char(0)
все якщо        все повторити
все повторити    повернути
newArray Все підпрограма

Підпрограма
    product(thirdArray,  size)
product:=1        повторити
для i від 1 до size        якщо
thirdArray[i]!=0
        то
            product*=int(thirdArray)
все якщо        product%=256
    повернути char(product) Все
підпрограма Блок-схема
алгоритму
```



Код програми

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

void generateFirstArray(char[], int); //генерує 1 масив
void generateSecondArray(char[], int); //генерує 2 масив
void generateThirdArray(char[], char[], char[], int); // генерує масив однакових елементів
char product(char[], int); //добуток елементів
void outputArray(char[], int); //виведення масиву

int main()
{
    const int size = 10;
    char firstArray[size], secondArray[size], thirdArray[size], result; //масиви та символьний результат
    //генерація масивів
    generateFirstArray(firstArray, size);
    generateSecondArray(secondArray, size);
    generateThirdArray(thirdArray, firstArray, secondArray, size);
    //виведення масивів
    cout << "First array:\n";
    outputArray(firstArray, size);
    cout << "Second array:\n";
    outputArray(secondArray, size);
    cout << "Third array:\n";
    outputArray(thirdArray, size);
    //обрахунок та виведення результату
    result = product(thirdArray, size);
    cout <<"Product: \n"<< result;

}

void generateFirstArray(char arr[], int size)
{
    for (int i = 0; i < size; i++) {
        arr[i] = 25 + 5 * (i+1);
    }
}

void generateSecondArray(char arr[], int size)
{
    for (int i = 0; i < size; i++) {
        arr[i] = 55 - 5 * (i + 1);
    }
}

void generateThirdArray(char newArr[], char firstArr[], char secondArr[], int size)
{
    bool match;
    for (int i = 0; i < size; i++) {
        match = false;
        for (int j = 0; j < size; j++) {
            if (firstArr[i] == secondArr[j])
                match = true;
        }
        if (match)
            newArr[i] = firstArr[i];
        else
            newArr[i] = 0;

    }
}
```

```
65
66  char product(char arr[], int size)
67  {
68      int  product=1;
69      for (int i = 0; i < size; i++) {
70          if (arr[i] != 0) {
71              product *= arr[i];
72          }
73      }
74      return product;
75  }
76
77  void outputArray(char arr[], int size)
78  {
79      for (int i = 0; i < size; i++) {
80          cout << arr[i]<<" ";
81      }
82      cout << "\n";
83  }
```



Випробування алгоритму

Блок	**Дія**
	Початок
1.	size:=10
2.	firstArray[size]=generateFirstArray(size)
3.	i=1,i<=10,i++
4.	newArray:=char(30)=’RS’
5.	i=2,i<=10,i++
6.	newArray:=char(35)=’#’

7.	i=3,i<=10,i++
8.	newArray:=char(40)=’(’
9.	i=4,i<=10,i++
10.	newArray:=char(45)=’-’
11.	i=5,i<=10,i++
12.	newArray:=char(50)=’2’
13.	i=6,i<=10,i++
14.	newArray:=char(55)=’7’
15.	i=7,i<=10,i++
16.	newArray:=char(60)=’<’
17.	i=8,i<=10,i++
18.	newArray:=char(65)=’A’
19.	i=9,i<=10,i++
20.	newArray:=char(70)=’F’
21.	i=10,i<=10,i++
22.	newArray:=char(75)=’K’
23.	i=11,i>=10
24.	firstArray:={RS,#,(,-,2,7,<,A,F,K}
25.	secondArray[size]=generateSecondArray(size)
26.	i=1,i<=10,i++
27.	newArray:=char(50)=’2’
28.	i=2,i<=10,i++
29.	newArray:=char(45)=’-’
30.	i=3,i<=10,i++
31.	newArray:=char(30)=’(’
32.	i=4,i<=10,i++
33.	newArray:=char(35)=’#’
34.	i=5,i<=10,i++
35.	newArray:=char(30)=’RS’
36.	i=6,i<=10,i++
37.	newArray:=char(25)=’EM’
38.	i=7,i<=10,i++
39.	newArray:=char(20)=’NAK’
40.	i=8,i<=10,i++
41.	newArray:=char(15)=’SI’

42.	i=9,i<=10,i++
43.	newArray:=char(10)=’LF’
44.	i=10,i<=10,i++
45.	newArray:=char(5)=’ENQ’
46.	i=11,i>=10

47.	secondArray:={2,-,(,#,RS,EM,NAK ,SI,LF,ENQ}
48.	thirdArray=generateThirdArray(firstArray,secondArray,size)
49.	i=1,i<=10,i++
50.	match=false
51.	j=1,j<=10,j++
52.	‘RS’!=’2’
53.	j=2,j<=10,j++
54.	‘RS’!=’-’
55.	j=3,j<=10,j++
56.	‘RS’!=’(’
57.	j=4,j<=10,j++
58.	‘RS’!=’#’
59.	j=5,j<=10,j++
60.	‘RS’==’RS’
61.	match:=true
62.	j=6,j<=10,j++
63.	‘RS’!=’EM’
64.	j=7,j<=10,j++
65.	‘RS’!=’NAK’
66.	j=8,j<=10,j++
67.	‘RS’!=’SI’
68.	j=9,j<=10,j++
69.	‘RS’!=’LF’
70.	j=10,j<=10,j++
71.	‘RS’!=’ENQ’
72.	j=11,j>10
73.	match==true
74.	newArray[1]=firstArray[1]
75.	i=2,i<=10,i++
76.	match=false
77.	j=1,j<=10,j++
78.	‘#’!=’2’
79.	j=2,j<=10,j++
80.	‘#’!=’-’
81.	j=3,j<=10,j++
82.	‘#’!=’(’
83.	j=4,j<=10,j++
84.	‘#’==’#’
85.	match:=true
86.	j=5,j<=10,j++

87.	‘#’!=’RS’
88.	j=6,j<=10,j++
89.	‘#’!=’EM’
90.	j=7,j<=10,j++
91.	‘#’!=’NAK’
92.	j=8,j<=10,j++
93.	‘#’!=’SI’
94.	j=9,j<=10,j++
95.	‘#’!=’LF’
96.	j=10,j<=10,j++
97.	‘#’!=’ENQ’
98.	j=11,j>10
99.	match==true
100	newArray[2]=firstArray[2]
101	i=3,i<=10,i++
102	match=false
103	j=1,j<=10,j++
104	‘(’!=’2’
105	j=2,j<=10,j++
106	‘(’!=’-’
107	j=3,j<=10,j++
108	‘(’==’(’
109	match:=true
110	j=4,j<=10,j++
111	‘(’!=’#’
112	j=5,j<=10,j++
113	‘(’!=’RS’
114	j=6,j<=10,j++
115	‘(’!=’EM’
116	j=7,j<=10,j++
117	‘(’!=’NAK’
118	j=8,j<=10,j++
119	‘(’!=’SI’
120	j=9,j<=10,j++
121	‘(’!=’LF’
122	j=10,j<=10,j++
123	‘(’!=’ENQ’
124	j=11,j>10
125	match==true
126	newArray[3]=firstArray[3]
127	i=4,i<=10,i++

128	match=false
129	j=1,j<=10,j++
130	‘-’!=’2’
131	j=2,j<=10,j++

132	‘-’==’-’
133	j=3,j<=10,j++
134	‘-’!=’(’
135	j=4,j<=10,j++
136	‘-’!=’#’
137	j=5,j<=10,j++
138	‘-’!=’RS’
139	j=6,j<=10,j++
140	‘-’!=’EM’
141	j=7,j<=10,j++
142	‘-’!=’NAK’
143	j=8,j<=10,j++
144	‘-’!=’SI’
145	j=9,j<=10,j++
146	‘-’!=’LF’
147	j=10,j<=10,j++
148	‘-’!=’ENQ’
149	j=11,j>10
150	match:=true
151	newArray[4]=firstArray[4]
152	i=5,i<=10,i++
153	match=false
154	j=1,j<=10,j++
155	‘2’==’2’
156	match:=true
157	j=2,j<=10,j++
158	‘2’!=’-’
159	j=3,j<=10,j++
160	‘2’!=’(’
161	j=4,j<=10,j++
162	‘2’==’#’
163	j=5,j<=10,j++
164	‘2’!=’RS’
165	j=6,j<=10,j++
166	‘2’!=’EM’
167	j=7,j<=10,j++

168	‘2’!=’NAK’
169	j=8,j<=10,j++
170	‘2’!=’SI’
171	j=9,j<=10,j++
172	‘2’!=’LF’
173	j=10,j<=10,j++
174	‘2’!=’ENQ’
175	j=11,j>10
176	match==true

177	newArray[5]=firstArray[5]
178	i=6,i<=10,i++
179	match=false
180	j=1,j<=10,j++
181	‘7’!=’2’
182	j=2,j<=10,j++
183	‘7’!=’-’
184	j=3,j<=10,j++
185	‘7’!=’(’
186	j=4,j<=10,j++
187	‘7’!=’#’
188	j=5,j<=10,j++
189	‘7’!=’RS’
190	j=6,j<=10,j++
191	‘7’!=’EM’
192	j=7,j<=10,j++
193	‘7’!=’NAK’
194	j=8,j<=10,j++
195	‘7’!=’SI’
196	j=9,j<=10,j++
197	‘7’!=’LF’
198	j=10,j<=10,j++
199	‘7’!=’ENQ’
200	j=11,j>10
201	match!=true
202	newArray[6]=char(0)=’NUL’
203	i=7,i<=10,i++
204	match=false
205	j=1,j<=10,j++
206	‘<’!=’2’
207	j=2,j<=10,j++

208	‘<’!=’-’
209	j=3,j<=10,j++
210	‘<’!=’(’
211	j=4,j<=10,j++
212	‘<’!=’#’
213	j=5,j<=10,j++
214	‘<’!=’RS’
215	j=6,j<=10,j++
216	‘<’!=’EM’
217	j=7,j<=10,j++
218	‘<’!=’NAK’
219	j=8,j<=10,j++
220	‘<’!=’SI’
221	j=9,j<=10,j++

222	‘<’!=’LF’
223	j=10,j<=10,j++
224	‘<’!=’ENQ’
225	j=11,j>10
226	match!=true
227	newArray[7]=char(0)=’NUL’
228	i=8,i<=10,i++
229	match=false
230	j=1,j<=10,j++
231	‘A’!=’2’
232	j=2,j<=10,j++
233	‘A’!=’-’
234	j=3,j<=10,j++
235	‘A’!=’(’
236	j=4,j<=10,j++
237	‘A’!=’#’
238	j=5,j<=10,j++
239	‘A’!=’RS’
240	j=6,j<=10,j++
241	‘A’!=’EM’
242	j=7,j<=10,j++
243	‘A’!=’NAK’
244	j=8,j<=10,j++
245	‘A’!=’SI’
246	j=9,j<=10,j++
247	‘A’!=’LF’

248	j=10,j<=10,j++
249	‘A’!=’ENQ’
250	j=11,j>10
251	match!=true
252	newArray[8]=char(0)=’NUL’
253	i=9,i<=10,i++
254	match=false
255	j=1,j<=10,j++
256	‘F’!=’2’
257	j=2,j<=10,j++
258	‘F’!=’-’
259	j=3,j<=10,j++
260	‘F’!=’(’
261	j=4,j<=10,j++
262	‘F’!=’#’
263	j=5,j<=10,j++
264	‘F’!=’RS’
265	j=6,j<=10,j++
266	‘F’!=’EM’

267	j=7,j<=10,j++
268	‘F’!=’NAK’
269	j=8,j<=10,j++
270	‘F’!=’SI’
271	j=9,j<=10,j++
272	‘F’!=’LF’
273	j=10,j<=10,j++
274	‘F’!=’ENQ’
275	j=11,j>10
276	match!=true
277	newArray[9]=char(0)=’NUL’
278	i=10,i<=10,i++
279	match=false
280	j=1,j<=10,j++
281	‘K’!=’2’
282	j=2,j<=10,j++
283	‘K’!=’-’
284	j=3,j<=10,j++
285	‘K’!=’(’
286	j=4,j<=10,j++
287	‘K’!=’#’

288	j=5,j<=10,j++
289	‘K’!=’RS’
290	j=6,j<=10,j++
291	‘K’!=’EM’
292	j=7,j<=10,j++
293	‘K’!=’NAK’
294	j=8,j<=10,j++
295	‘K’!=’SI’
296	j=9,j<=10,j++
297	‘K’!=’LF’
298	j=10,j<=10,j++
299	‘K’!=’ENQ’
300	j=11,j>10
301	match!=true
302	newArray[10]=char(0)=’NUL’
303	i=11,i>10
304	thirdArray={RS,#,(,-,2,NUL, NUL, NUL , NUL , NUL}
305	result:=product(thirdArray, size)
306	product:=1
307	i=1,i<=10,i++
308	‘RS’!=’NUL’
309	product*=30=30
310	i=2,i<=10,i++
311	‘#’!=’NUL’
312	product*=35=1050
313	i=3,i<=10,i++
314	‘(‘!=’NUL’
315	product*=40=4200
316	i=4,i<=10,i++
317	‘-‘!=’NUL’
318	product*=45=1890000
319	i=5,i<=10,i++
320	‘2’!=’NUL’
321	product*=50=94500000
322	i=6,i<=10,i++
323	‘NUL’ ==’NUL’
324	i=7,i<=10,i++
325	‘NUL’ ==’NUL’
326	i=8,i<=10,i++
327	‘NUL’ ==’NUL’
328	i=9,i<=10,i++

329	‘NUL’==’NUL’
330	i=10,i<=10,i++
331	‘NUL’==’NUL’
332	i=11,i>10
333	product%=256=160
334	result:=char(160)=á
335	**Виведення** á
	Кінець

Висновки

Протягом виконання цієї лабораторної роботи я набув навичок використання методів послідовного пошуку у невпорядкованих послідовностях та практичних навичок їх використання під час складання програмних специфікацій. Маючи формули задання елементів двох одновимірних символьних масивів, я склав програму яка успішно знаходить масив, який складається з спільних елементів попередніх масивів та добуток його ненульових елементів.