МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 5

з дисципліни
«Алгоритмізація та програмування»

Виконав:
студент групи КН-108
Мокрик Ярослав
Викладач:
Гасько Р. Т.

Львів – 2018 р.

Тема: "Функції і масиви"

Мета: Організувати обробку масивів з використанням функцій, навчитися передавати масиви як параметри функцій.

1. Короткі теоретичні відомості

1.1. Функції

Функцію в С можна розглядати:

- як один з похідних типів даних (поряд з масивами й вказівниками);
- як мінімальний виконавчий модуль програми (підпрограму).

Всі функції мають єдиний формат визначення:

<тип><ім'я_функції>(<список_формальних_параметрів>) , де

<тіло_функції> , де

<тип> - або void, якщо функція не повертає значення, або тип значення, що повертається функцією,;

<ім'я_функції> - або main для головної функції, або довільний ідентифікатор, що не збігається зі службовими словами й іменами інших об'єктів програми;

<список_формальних_параметрів> - або порожній ( ), або список, кожен елемент якого має вигляд:

<позначення_типу><ім'я_параметра>

Наприклад:

(int k)

(char i, char j, int z)

<тіло_функції> - це частина визначення функції, взята у фігурні дужки { }. Тіло функції може бути або складеним оператором, або блоком. Визначення функцій не можуть бути вкладеними.

Для передачі результату з функції у викликаючу функцію використовується оператор return. Він може використовуватися у двох формах:

1) return; - завершує функцію, яка не повертає жодного значення (тобто перед ім'ям функції зазначений тип void)
2) return <вираз>; - повертає значення виразу, вираз повинне мати тип, зазначений перед ім'ям функції.

Якщо програміст не пише оператор return явно, то компілятор автоматично дописує return у кінець тіла функції перед закриваючою фігурною дужкою “}”.

Приклад:

```
int op (char c, int x, int  y)
{
switch c
{
case '+' : return x+y;
case '-' : return x-y;
case '*' : return x*y;
case '/' : return x/y;
default: cout<<“\nоперация не визначена”;return 0;
}
}
```

Виклик функції здійснюється в такий спосіб:

<позначення функції>(<список фактичних параметрів>); де

<позначення функції> - або ім'я функції, або вказівник на функцію;

<список фактичних параметрів> - список виразів, кількість яких дорівнює числу формальних параметрів функції. Між формальними й фактичними параметрами повинна бути відповідність по типах.

Наприклад:

```
c = op ( '+', 5 ,4 );
```

Синтаксис С передбачає тільки один спосіб передачі параметрів - передача за значенням (тобто змінити значення параметрів всередині функції не можна). Але існує можливість опосередковано змінити значення змінних переданих у вигляді параметрів: за допомогою вказівника у функцію, яка викликається можна передати адресу будь-якого об'єкта із викликаючої програми. Якщо вказівник розіменувати, то отримаємо значення, записане за цією адресою.

Приклад:

1)

```
//опис функції для обміну змінних a й b
void change (int a,int b)
{
int r;
r = a; a = b; b = r;
}
// виклик функції
change(a, b);
```

Обміну не відбудеться, тому що результат не буде переданий у викликаючу програму.

2)

```
void change (int *a,int *b)
{
```

```
int r;
r = *a; *a = *b; *b = r;
}
// виклик функції
change(&a, &b);
```

При виклику передаються адреси, за якими перебувають значення й виконується обмін значень, які перебувають за цими адресами.

1.2. Масиви й рядки як параметри функцій

Якщо в ролі параметр функції використовується позначення масиву, то насправді у функцію передається адреса першого елемента масиву.

Приклад:

```
//обчислення суми елементів масиву
//варіант 1
int sum (int n, int a[] )
{
int i,int s=0;
for( i=0; i<n; i++ )
s+=a[i]
return s;
}
void main()
{
int a[]={ 3, 5, 7, 9, 11, 13, 15 };
int s = sum( 7, a );
cout<<s;
}
```

```
//варіант 2
int sum (int n, int *a)
{
for(int i=0, s=0; i<n; s+=*(a+i),i++ );
return s;
}
void main()
{
int a[]={ 3, 5, 7, 9, 11, 13, 15 };
 int s = sum( 7, a );
cout<<s;
}
```

Рядки в ролі фактичних параметрів можуть бути визначені або як одновимірні масиви типу char[], або як вказівники типу char*. На відміну від звичайних масивів у цьому випадку немає необхідності явно вказувати довжину рядка.

Постановка завдання

Використовуючи функції, розв’язати зазначене у варіанті завдання. Масив повинен передаватися у функцію як параметр.

Варіант завдання

19.

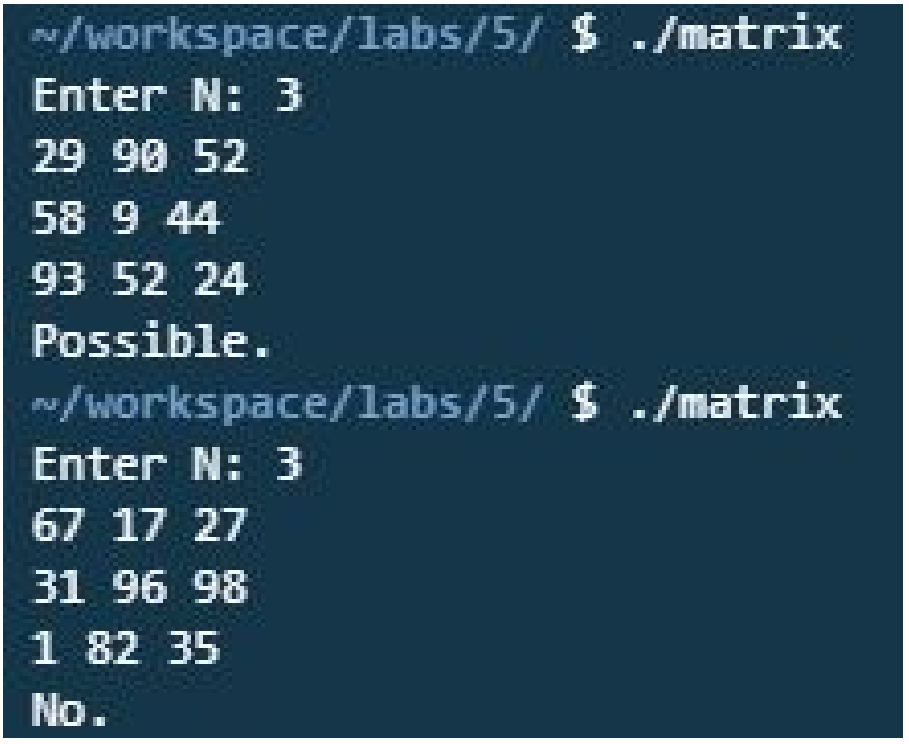
Задано рядок з N^2 цифр. Встановити чи можна, розбивши рядок на підстрічки довжиною N, записати їх у рядки двовимірного масиву N x N по одній цифрі в одному елементі так, щоб вони в першому стовпці розташувалися в порядку зростання.

Текст програми

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

bool isGood (int n, int nums[][n]) {
    bool possible = true;
    for (int i = 0; i < n-1; i ++) {
        printf("%d\n", nums[i][0]);
        if  (nums[i][0] > nums[i+1][0])
            possible = false;
    }
    printf("%d\n", nums[n-1][0]);
    return possible;
}

int main() {
    //Initialize random number generator
    time_t t;
    srand((unsigned) time(&t));

    int n = 0;
    printf("Enter N: ");
    scanf("%d", &n);
    int numbers[n*n];
    for (int i = 0; i < n*n; i ++) {
        numbers[i] = rand()%100;
    }
    int matrix[n][n];
    for (int i = 0; i < n; i ++) {
        for (int j = 0; j < n; j ++) {
```

```
            matrix[i][j] = numbers[i * n + j];
        }
    }
    if (isGood(n, matrix))
        printf("Possible.\n");
    else
        printf("No.\n");
}
```

Результати



29 < 58 < 93

Так, можливо.

67 > 31 > 1

Ні, неможливо.