

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
до лабораторної роботи №4
із дисципліни «Програмування»
на тему
УКАЗІВНИКИ, ФУНКЦІЇ, РЕКУРСІЯ

Виконав:
студент групи КМ-93
Пиндиківський Т. Р.

Керівник:
асистент Дрозденко О. М.

ЗМІСТ

МЕТА РОБОТИ	3
ПОСТАНОВКА ЗАДАЧІ	4
ОСНОВНА ЧАСТИНА	5
ВИСНОВКИ	9
ДОДАТОК А	10
ДОДАТОК Б	14

МЕТА РОБОТИ

Ознайомитися з новим типом організації даних С (вказівники) і набути практичних навичок його використання для написання програм; оволодіти синтаксисом написання функцій і їхніх прототипів, методикою складання і відлагодження програм, що містять функції, специфікою передачі параметрів у функцію та повернення одержаних результатів; вивчити методи використання алгоритмів і програм із рекурсією мовою С.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1:

1. Обчислити за допомогою функції кількість додатних, від'ємних і нульових елементів різних матриць.

Завдання 2:

2. Написати рекурсивну функцію знаходження цифрового кореня натурального числа. Цифровий корінь натурального числа отримують у такий спосіб. Якщо скласти всі цифри цього числа, потім всі цифри знайденої суми і повторювати цей процес, то в результаті буде отримано однозначне число (цифра), яка і називається цифровим коренем даного числа.

ОСНОВНА ЧАСТИНА

Завдання 1 :

Для виконання програми спочатку імпортуються наступні модулі :

- ***stdio.h*** – файл заголовку для стандартних операцій введення/виведення;
- ***time.h*** – бібліотека мови C, що містить функціонал для генерації довільних чисел;
- ***stdlib.h*** – стандартна бібліотека мови C.
- ***isnumber.h*** – бібліотека, що містить функції для обробки даних, введених користувачем.

Змінні, що використовуються під час виконання завдання:

- ***rows* (int)** – ціла величина, в якій зберігається значення кількості рядків масиву;
- ***cols* (int)** – цілочисельна величина, що дорівнює кількості стовпців масиву;
- ***i, j* (int)** – допоміжні числа для ітерації по елементах масиву;
- ***choice* (int)** – ціле значення, що визначає режим заповнення масиву числами;
- ***array*[][] (float)** – масив чисел;
- ***prompt*[] (char)** – рядкова величина для підказки про введення даних користувачем.
- ****pointer* (float)** – вказівник на елементи масиву *array*
- ***counter_positive, counter_negative, counter_zeros* (int)** – “лічильники” для підрахунку додатніх, від’ємних та нульвих значень масиву.

Хід виконання завдання:

1. Вводиться значення кількості рядків та стовпців масиву з використанням власної функції ***input_int_positive_number(prompt)***, яка повертає введене користувачем ціле число (детальніше у додатку Б).

rows=input_int_positive_number("\n\nEnter the number of rows in a 2d array");

cols=input_int_positive_number("\n\nEnter the number of cols in a 2d array : ");

2. Потім аналогічно вводиться з консолі значення змінної *choice*, що визначає метод заповнення масиву числами вручну або довільними числами (з використанням функції *input_int_positive_number*):

```
choice=input_int_positive_number("\n\nPress 1 - to fill array manually, 2 - to fill with random numbers : ");
```

3. При виборі опції заповнення масиву довільними значеннями

«запускається» два цикли *for()* із подальшою ітерацією по значенню змінної *i* (від 0 до («кількість рядків»-1)) та *j* (від 0 до («кількість стовпців»-1)) та присвоєнню кожному елементу масиву *array[i][j]* довільного значення, що отримується при виклику функції *rand()*, яка повертає довільне число з вказаного діапазону.

4. При виборі опції заповнення масиву вручну «запускається» два цикли *for()* із подальшою ітерацією по значенню змінної *i* (від 0 до («кількість рядків»-1)) та *j* (від 0 до («кількість стовпців»-1)) та присвоєнню кожному елементу масиву *array[i][j]* значення, введеного користувачем із застосуванням функції *input_int_positive_number()* (детальніше у додатку Б).

5. Сформувавши цикл *for()* із подальшою ітерацією по значенню вказівника **pointer* , що вказує на адресу кожного наступного елемента масиву, відбувається збільшення лічильників *counter_positive*, *counter_negative*, *counter_zeros* на одиницю у випадку, коли значення вказівника додатне/менше нуля/нуль.

6. В кінці з використанням функцій *printf()* із специфікаторами типів для цілих значень ("*%d*"), у консоль виводиться відповідна інформація про кількість додатних, від'ємних і нульових елементів масиву:

```
printf("\n\nThe matrix contains %d positive numbers.", counter_positive);  
printf("\n\nThe matrix contains %d negative numbers.", counter_negative);  
printf("\n\nThe matrix contains %d zero numbers.", counter_zeros);
```

Завдання 2 :

Для виконання програми спочатку імпортуються наступні модулі :

- ***stdio.h*** – файл заголовку для стандартних операцій введення/виведення;
- ***time.h*** – бібліотека мови C, що містить функціонал для генерації довільних чисел;
- ***stdlib.h*** – стандартна бібліотека мови C.
- ***isnumber.h*** – бібліотека, що містить функції для обробки даних, введених користувачем.

Змінні, що використовуються під час виконання завдання:

- ***rows (int)*** – ціла величина, в якій зберігається значення кількості рядків масиву;
- ***cols (int)*** – цілочисельна величина, що дорівнює кількості стовпців масиву;
- ***i (int)*** – допоміжне число для ітерації по елементах масиву;
- ***choice (int)*** – ціле значення, що визначає режим заповнення масиву числами;
- ***array[] (float)*** – масив чисел;
- ***prompt[] (char)*** – рядкова величина для підказки про введення даних користувачем.
- ***sum (int)*** – сума цифр числа;
- ***number (int)*** – ціле число, введене користувачем.
- ***digital_root (int)*** – рекурсивна функція для обчислення

Хід виконання завдання:

1. Вводиться значення числа ***number*** з умови задачі з використанням власної функції ***input_int_positive_number(prompt)***, яка повертає введене користувачем ціле число (детальніше у додатку Б).

```
number=input_int_positive_number("\n\nEnter a positive integer (natural) number : ");
```

2. Потім, якщо число є меншим 10, то його значення виводиться у консоль з використанням функції *printf()*:

```
printf("\n\nThe digital root of number %d is %d", number, digital_root(number));
```

3. У іншому випадку до значення змінної *sum* додається циклічно остача від ділення числа *number* на 10 та відбувається ділення *number* на 10, допоки значення змінної *number* є більшим за нуль. Ці команди рекурсивної функції *digital_root()* викликаються допоки на вхід функція не отримає одноцифрове число, яке і є результатом виконання завдання та буде повернене функцією.

4. В кінці виводиться цифровий корінь числа *number* – результат виконання функції *digital_root()* із застосуванням функції *printf()* та специфікатором типу “%d”.

ВИСНОВКИ

На цій лабораторній роботі було ознайомлено з новим типом організації даних С (вказівники) і набуто практичні навички їх використання для написання програм; був вичений синтаксис для написання функцій і їхніх прототипів, методики складання і відлагодження програм, що містять функції, специфікою передачі параметрів у функцію та повернення одержаних результатів; вивчено методи використання алгоритмів і програм із рекурсією мовою С.

ДОДАТОК А

1. Програмна реалізація задачі №1

```
#include<stdio.h>
#include<time.h>
#include<string.h>
#include<stdlib.h>
#include<stdbool.h>
//#include"isnumber.h"
void program1 ()
{

    //    srand(time(0));

    int rows, cols, i, j;

    rows=input_int_positive_number("\n\nEnter the number of rows in a 2d
array : ");

    cols=input_int_positive_number("\n\nEnter the number of cols in a 2d
array : ");

    int choice;

    do
    {

        choice=input_int_positive_number("\n\nPress 1 - to fill array
manually, 2 - to fill with random numbers : ");

        } while(choice<0 || choice>2);

    //    char number_chr[100];

    float array[rows][cols];

    for (i=0; i<rows; i++)
        for (j=0; j<cols; j++)
            array[i][j]=0;

    int f, k;
    //    bool result=true;
    char prompt[100];
    //char array[10];
    //sprintf(array, "%f", 3.123);

    char data[100];

    switch(choice)
    {
        case 1:
        {
```

```

        for (i=0; i<rows; i++)
        {
            for (j=0; j<cols; j++)
            {
                do
                {
                    printf("\n\nEnter the value of number in
%d row and %d column : ", i+1, j+1);
                    gets(data);
                    if(!(is_number(data)))
                        printf("\nYou have entered wrong
value");
                } while((is_number(data)!=true));

                array[i][j]=atof(data);

            }
        }
        printf("\n\nArray with manually entered numbers :\n");
        for (i=0; i<rows; i++)
        {
            printf("\n");
            for (j=0; j<cols; j++)
                printf(" %.2f ", array[i][j]);
            printf("\n");
        }
        break;
    }

    case 2:
    {
        printf("\n\nArray with random numbers :\n");
        for (i=0; i<rows; i++)
        {
            printf("\n");
            for (j=0; j<cols; j++)
            {
                array[i][j]= rand () % (rows*cols) - (rand() %
(rows*cols)/2);

                printf(" %.2f ", array[i][j]);
            }
            printf("\n");
        }
        break;
    }

}

float *pointer;
pointer=&array[0][0];
int counter_negative=0, counter_positive=0, counter_zeros=0;

for(i=0;i<rows*cols; pointer++, i++)
{
    if(*pointer<0)
        counter_negative++;
    else if(*pointer>0)
        counter_positive++;
    else if((int)*pointer==0)
        counter_zeros++;
}

```

```

    }

    printf("\n\nThe matrix contains %d positive numbers.",
counter_positive);
    printf("\n\nThe matrix contains %d negative numbers.",
counter_negative);
    printf("\n\nThe matrix contains %d zero numbers.", counter_zeros);
}

```

```

Task 1 : Count the number of positive, negative, and zero numbers in the matrix.
Enter the number of rows in a 2d array : 3

Enter the number of cols in a 2d array : 3

Press 1 - to fill array manually, 2 - to fill with random numbers : 2

Array with random numbers :

1.00  5.00  8.00

3.00  6.00  7.00

4.00  5.00  3.00

The matrix contains 9 positive numbers.
The matrix contains 0 negative numbers.
The matrix contains 0 zero numbers.

If you want to continue testing program, press c button ...

```

Рис.1 – Тестування завдання №1

2. Програмна реалізація задачі №2

```

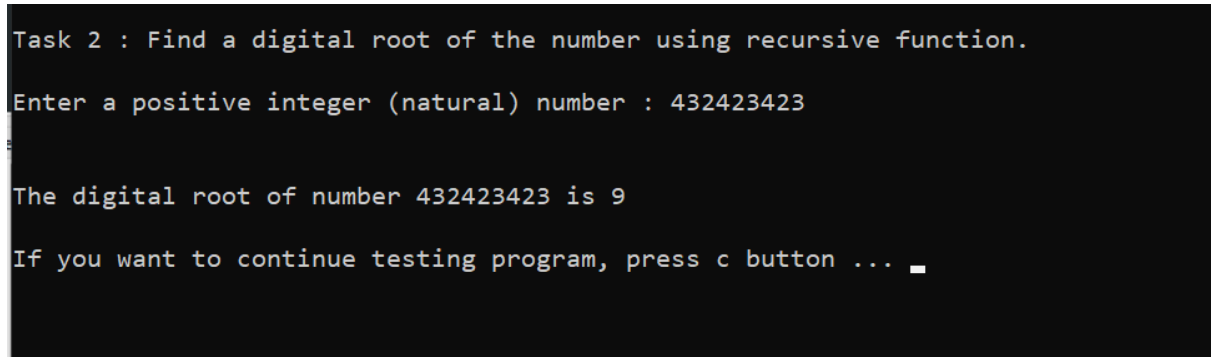
#include<stdio.h>
#include<time.h>
#include<string.h>
#include<stdlib.h>
//#include"isnumber.h"
long int sum_ciphers(long int number)
{
    long int sum=0, bufer=number;

```

```

        while(number>0)
        {
            sum+=number%10;
            number=(int) (number/10);
        }
        return sum;
    }
    long int digital_root(long int number)
    {
        if(number>=0 && number<=9)
            return number;
        else
            digital_root(sum_ciphers(number));
    }
    void program2()
    {
        long int number=0;
        number=input_int_positive_number("\n\nEnter a positive integer
(natural) number : ");
        printf("\n\nThe digital root of number %d is %d", number,
digital_root(number));
    }
}

```



```

Task 2 : Find a digital root of the number using recursive function.
Enter a positive integer (natural) number : 432423423
The digital root of number 432423423 is 9
If you want to continue testing program, press c button ... _

```

Рис.2 – Тестування завдання №2

ДОДАТОК Б

Функція *input_int_positive_number(prompt)* приймає від користувача введення даних цілого типу наступним чином : через функцію *gets()*, що приймає з консолі рядкову величину у тілі циклу *while()*, умовою виходу з якого є отримання рядка, що відповідає цілочисельній величині. Потім відбувається конвертація отриманих рядкових даних у число з використанням функції *atoi()* та повертається функцією *input_int_positive_number(prompt)* як результат виконання.