НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №*4*

із дисципліни «Основи програмування»

на тему

«Указівники, функції, рекурсія»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-73 | *асистент Громова В.В.* |
| *Садченко М.В.* |  |

Київ — 2018

ЗМІСТ

[ВСТУП 3](#_Toc509382623)

[1 ПОСТАНОВКА ЗАДАЧІ 3](#_Toc509382624)

[1.1 Мета роботи 3](#_Toc509382625)

[1.2 Варіант та завдання лабораторної роботи 3](#_Toc509382626)

[2 ОПИС КОДУ 4](#_Toc509382627)

[2.1 Опис першої програми 4](#_Toc509382628)

[2.2 Опис другої програми 4](#_Toc509382629)

[2.3 Опис бібліотек 5](#_Toc509382630)

[ВИСНОВКИ 5](#_Toc509382631)

[ПЕРЕЛІК ПОСИЛАНЬ 5](#_Toc509382632)

[ДОДАТОК А (Відповіді на контрольні запитання) 6](#_Toc509382633)

[ДОДАТОК Б (Код програми) 8](#_Toc509382634)

[Перша програма 8](#_Toc509382635)

[Друга програма 10](#_Toc509382636)

[ДОДАТОК В (Результат роботи програми) 12](#_Toc509382637)

[Перша програма 12](#_Toc509382638)

[Друга програма 12](#_Toc509382639)

[ДОДАТОК Г (Блок-схема) 13](#_Toc509382640)

# ВСТУП

Даний звіт створено для детального ознайомлення з лабораторною роботою №4, її завданням, структурою, розробкою програмного коду мовою програмування C, у супроводі детального розбору поставлених вимог.

# 1 ПОСТАНОВКА ЗАДАЧІ

## 1.1 Мета роботи

Ознайомитися з новим типом організації даних С (указівники) 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.

## 1.2 Варіант та завдання лабораторної роботи

Завдання з варіанту №24:

1. Дана послідовність натуральних чисел (одне число в рядку), що закінчується двома числами 0 підряд. Визначте, скільки разів у цій послідовності зустрічається число 1. Числа, що йдуть після двох нулів, необхідно ігнорувати.
2. Магічним квадратом порядку n називається квадратна матриця розміром n x n, що складається з чисел 1, 2, ..., n2 так, що суми по кожному стовпцю, кожному рядку і кожній з двох діагоналей рівні між собою. Написати функцію генерації магічного квадрату третього порядку (у якого по три стовпці і рядки). Можна скористатися тим фактом, що число (вираз), що стоїть на перетині його діагоналей, завжди дорівнює ⅓ магічної константи, яка обчислюється за такою формулою: (n\*(n2+1))/2. Для матриці 3х3 це число дорівнює 15. Будувати будь-який непарний квадрат потрібно починати з центральної комірки, в яку записати 1.

# 2 ОПИС КОДУ

## 2.1 Опис першої програми

Програма починається з того, що користувачеві надається вибір між доступом до самовільного введення числової послідовності (кнопка «1»), так званим рандомом, що сам генерує послідовність (кнопка «2»), очисткою консолі («8») та виходом з програми («9»). Реалізовано за допомогою циклу do-while, функцій strchr (перевіряє, чи входить введений символ в заданий масив) та getch (<conio.h>) (за її допомогою компьютер отримує символ одразу після натискання; користувачеві непотрібно підтверджувати свій вибір Enter`ом). Далі правильний символ надходить до switch-case, що розмежує кожну з дій окремою підпрограмою. Для того, щоб з одного ‘кейсу’ програма просто не переходила у інший – в кінці кожної з підпрограм ставиться break.

Очистка консолі виконується за допомогою команди system(“cls”) з бібліотеки <stdlib.h>.

Вихід з програми реалізовано командою exit(0), теж з <stdlib.h>.

Перша програма починається з того, що користувач вводить число, що являє собою кількість чисел у послідовності. Звичайно, кожний ввід контролюється, щоб не виникло помилки. У цьому випадку – це ‘валідація’ натуральних чисел, що знаходиться в окремій функції. Для ‘валідації’ використовуються, окрім циклу while, ще такі функції, як strlen() (рахує кількість елементів у масиві), isdigit() (перевіряє числове значення(чи є елемент цифрою)) та atoi() (конвертує строковий тип до цілих чисел).

Далі вже користувач вводить самі числа, якщо він натиснув «1». Якщо ж його вибір зупинився на кнопці «2», то за нього числа підбирає комп’ютер за допомогою псевдовипадкових чисел (функції rand() та srand()). Користувачеві необхідно вказати лише їх діапазон. Таким чином проходило заповнення масиву у окремій функціїї ‘filling()’.

Коли всі числа введені, масив наповнено, то комп’ютер у функції ‘counting()’ починає підраховувати всі одиниці до перших двух нулів. Далі це число одиниць виводиться користувачеві на екран.

Після закінчення кожної програми (окрім виходу з програми) відбувається рекурсивний повтор, що дозволяє ще раз обрати між заданими діями, натиснувши «1», «2», «8» чи «9». У всіх функціях повторювані дії здійснювалися за допомогою рекурсії.

## 2.2 Опис другої програми

Друга програма має ту ж саму структуру, що й перша. Тобто з головної функції здійснювалися переходи до інших. Користувачеві теж надається вибір між виконанням головної підпрограми («1»), очисткою консолі («8»), виходом з програми («9»).

Натиснувши першу кнопку, користувач у силах обрати НЕпарне число, що означатиме початок обчислення програмою «магічного квадрату» заданої довжини. Цей квадрат міститиме всі числа, починаючи з одиниці, закінчуючи n^2 (n – довжина).

Коли буде побудований двовимірний масив, користувачеві виведеться на екран сума кожного рядка, стовпчика, діагоналі (це все одне число), та сам «магічний квадрат» у вигляді матриці з неповторюваними числами.

## 2.3 Опис бібліотек

Для початку, опишемо використані бібліотеки у програмі:

<stdio.h> - відповідає за ввод\вивід даних;

<stdlib.h> - відповідає за функції, що займаються виділенням пам'яті, контроль процесу виконання програми, перетворення типів та інші;

<string.h> - містить функції для роботи з нуль-термінованими рядками і різними функціями роботи з пам'яттю;

<ctype.h> - містить оголошення функцій для класифікації символів;

<conio.h> - застосовується для створення текстового інтерфейсу;

<time.h> - містить типи та функції для роботи з датою та часом.

# ВИСНОВКИ

Під час виконання лабораторної роботи були вивчені нюанси роботи із функціями, вказівниками у мові програмування С. Також у всіх функціях було застосовано рекурсію, що дозволило нам краще засвоїти цю тему.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Програмування мовою С. Методичні вказівки до виконання лабораторних робіт із кредитного модуля «Програмування-2. С.» для студентів спеціальності 113 «Прикладна математика» [Текст] / В. В. Громова. — К. : НТУУ «КПІ», 2016. —5- 62 с.

# ДОДАТОК А (Відповіді на контрольні запитання)

1. *Що таке вказівник?*

Це адреса елемента пам’яті, яка розподіляється для розміщення деякого об’єкта (як такий об’єкт може виступати змінна, масив, структура, рядкова константа).

1. *Спільне між указівником і масивом, їхні взаємні переваги й вади.*

У мові С між указівниками й масивами існує тісний зв’язок. Наприклад, оголошення масиву int array[25] визначає не тільки виділення пам’яті для двадцяти п’яти елементів масиву, але й для вказівника з іменем array. Ім’я масиву без індексів трактується як адреса початкового елемента. Тобто ім’я масиву є вказівником на масив. Таким чином, доступ до елементів масиву здійснюється через указівник з іменем array.

Оскільки ім’я масиву є вказівником, допустиме таке присвоєння:

int arrаy[25];

int\* ptr;

ptr = array;

У цьому фрагменті вказівник ptr установлюється на адресу першого елемента масива, при цьому присвоювання ptr = arrаy можна записати в еквівалентній формі ptr = &arrаy[0].

1. *Опис функції. Для чого оголошують прототипи?.*

*Функція* — це деяка логічно завершена сукупність операторів мови, яка виконує певну задачу, і яку можна викликати для виконання потрібну кількість разів. Основна форма опису функції має вигляд:

*<тип> <ім’я функції>* (*<формальні параметри>*) {*<тіло функції>*}

де *<тип>* визначає тип значення, яке повертає функція за допомогою оператора return.

*Прототип функції* можна вказувати замість її опису до моменту її виклику для того, щоб компілятор міг виконати перевірку відповідності типів аргументів і параметрів. Прототип функції за формою такий же, як і заголовок функції. Наприкінці нього ставлять ;.

1. *Що таке формальні й фактичні параметри? Локальні та глобальні?*

*<формальні параметри>* — список параметрів функції, який складається з переліку типів та імен параметрів, розділених комами. Функція може не мати параметрів, але круглі дужки в її описі неодмінні в будь-якому випадку.

Параметри, які записують у зверненні до функції, називають *фактичними*, а параметри, зазначені в описі функції — *формальними*. Фактичні параметри повинні відповідати формальним за кількістю, порядком слідування й типом. Об’єкти, оголошені поза функцією, діють у будь-якій функції. Їх називають *глобальними*. Об’єкти, оголошені всередині функції, діють тільки в ній. Їх називають *локальними*.

1. *Яким чином передати функції одновимірний масив як параметр? Багатовимірний масив? Рядок?*

Існує три способи оголошення параметра, який буде отримувати вказівник масиву:

* func(int ar[10]); — оголошення як масиву;
* func(int ar[]);— оголошення як безрозмірного масиву;
* func(int\* ar); — оголошення як указівника.

Усі три способи породжують один і той же результат — указівник.

*Двовимірні масиви як аргументи функції*

У випадку використання як аргумента функції багатовимірного масиву найліпше вказувати масив з усіма його розмірами, наприклад:

void mult(int a[5][5], int b[5][5], int c[5][5]);

1. *Чим відрізяються функції типу* void*?*

Вони не повертають ніякого значення.

1. *Що таке рекурсія?*

*Рекурсія* — це спосіб організації допоміжного алгоритму у вигляді підпрограми (процедури або функції), за якого вона в ході виконання звертається сама до себе. Рекурсивним називають будь-який об’єкт, який частково визначається через себе.

1. *Як у рекурсії змінюються локальні й глобальні змінні?*

Коли функція викликає саму себе, комп’ютер розподіляє пам’ять для нових локальних змінних і параметрів у стеку та виконує код функції з цими новими змінними спочатку.

Рекурсивне звернення не створює нової копії функції. Новими є тільки аргументи. По мірі того, як кожне рекурсивне звернення повертається, комп’ютер видаляє зі стека старі локальні змінні й параметри та відновлює виконання з точки виклику функції всередині функції.

1. *Що таке гранична умова, і яке її призначення в рекурсивній функції?*

Гранична умова – це метод запобігання вічного циклу та вчасного виходу з рекурсії.

1. *Що таке рекурсивний спуск?*

Породження все нових копій рекурсивної функції до виходу на граничну умову називають *рекурсивним спуском*.

1. *Що таке рекурсивний підйом?*

Завершення роботи рекурсивних функцій, аж до найпершої, яка ініціювала рекурсивні виклики, називають *рекурсивним підйомом*.

1. *Чи передбачено автоматичний захист пам’яті для рекурсивних функцій у випадку переповнення стека обчислень?*

Ні. За кожного нового рекурсивного звернення до пам’яті створюється нова копія локальних змінних. В разі відсутності граничної умови необмежене зростання числа таких копій призведе до аварійного завершення програми через переповнення стека.

# ДОДАТОК Б (Код програми)

## Перша програма

#include <stdio.h>

#include <string.h> // strlen(), strchr()

#include <stdlib.h> // atoi(), exit(0), system("cls")

#include <ctype.h> // isdigit()

#include <conio.h> // getch()

#include <time.h> // time()

int valid\_int() {

char str[65];

int num;

int smb = 0;

scanf("%s", str); *//input of value*

int len = strlen(str);

for (int i=0; i<len; ++i) {

if (isdigit(str[i]) == 0) {

++smb;

**break**;

}

}

if (smb == 0) {

num = atoi(str); *//convert str to int*

return num;

}

else printf("need natural number!: ");

return valid\_int();

}

int counting(int c, int n, int count, int ar[]) {

if (c > n-1) {

printf("Counts of '1' before two zeros: %d**\n**", count);

return 0;

}

if ((ar[c] == 0) && (c < n-1)) {

if (ar[c+1] == 0) {

for (int j = 0; j < c; ++j) {

if (ar[j] == 1) {

++count;

}

}

}

}

return counting(c+1, n, count, ar);

}

int filling(int c, int n, int ar[]) {

if (c > n-1 ) {

return counting(0, n, 0, ar);

}

printf("%d) ", c+1);

ar[c] = valid\_int();

return filling(c+1, n, ar);

}

int random(int c, int n, int max\_ran, int ar[]) {

if (c > n-1 ) {

return counting(0, n, 0, ar);

}

ar[c] = rand()%max\_ran;

printf("array[%d] = %d**\n**", c, ar[c]);

return random(c+1, n, max\_ran, ar);

}

int main() {

int choice;

fflush(stdin);

printf("**\n**Press '1' to input sequence.**\n**"\

"Press '2' to get random sequence.**\n**"\

"Press '8' to clear console.**\n**"\

"Press '9' to get out program.**\n**");

do {

choice = getch();

} while(!strchr("1289", choice));

switch (choice) {

case '1': {

printf("**\n**Number of elements: ");

int n = valid\_int();

int ar[n];

printf("**\n**Enter natural sequence**\n**");

filling(0, n, ar);

**break**;

}

case '2': {

time\_t t;

*// Intializes random number generator*

srand((unsigned) time(&t));

printf("**\n**Enter max range: ");

int max\_ran = valid\_int();

printf("Number of elements: ");

int n = valid\_int();

int ar[n];

random(0, n, max\_ran, ar);

**break**;

}

case '8': {

printf(" 7 ballov :)**\n**");

system("cls");

**break**;

}

case '9': exit(0); **break**;

default: **break**;

}

return main();

}

## Друга програма

#include <stdio.h>

#include <string.h> // strlen(), strchr()

#include <stdlib.h> // atoi(), exit(0), system("cls")

#include <ctype.h> // isdigit()

#include <conio.h> // getch()

int valid\_odd() {

char str[65];

int num;

int smb = 0;

scanf("%s", str); *//input of value*

int len = strlen(str);

for (int i=0; i<len; ++i) {

if (isdigit(str[i]) == 0) {

++smb;

**break**;

}

}

if (smb == 0) {

num = atoi(str); *//convert str to int*

if (num%2 == 1) return num;

else printf("need odd number!: ");

}

else printf("need natural number!: ");

return valid\_odd();

}

void generateSquare(int n) {

int magicSquare[n][n];

*// set all slots as 0*

memset(magicSquare, 0, sizeof(magicSquare));

*// Initialize position for 1*

int i = n/2;

int j = n-1;

for (int num=1; num <= n\*n; ) {

if (i < 0 && j == n) {

j = n-2;

i = 0;

}

else {

if (j == n) j = 0;

if (i < 0) i = n-1;

}

if (magicSquare[i][j]) {

j -= 2;

i++;

continue;

}

else

magicSquare[i][j] = num++;

j++; i--;

}

printf("Sum of each row or column - %d**\n\n**", n\*(n\*n+1)/2);

for (i=0; i<n; i++) {

for (j=0; j<n; j++)

printf("%3d ", magicSquare[i][j]);

printf("**\n**");

}

}

int main() {

int choice;

fflush(stdin);

printf("**\n**Press '1' to get Magic Square.**\n**"\

"Press '8' to clear console.**\n**"\

"Press '9' to get out program.**\n**");

do {

choice = getch();

} while(!strchr("189", choice));

switch (choice) {

case '1': {

printf("**\n**The Magic Square for n = ");

int n = valid\_odd();

generateSquare(n);

**break**;

}

case '8': {

printf(" 7 ballov :)**\n**");

system("cls");

**break**;

}

case '9': exit(0); **break**;

default: **break**;

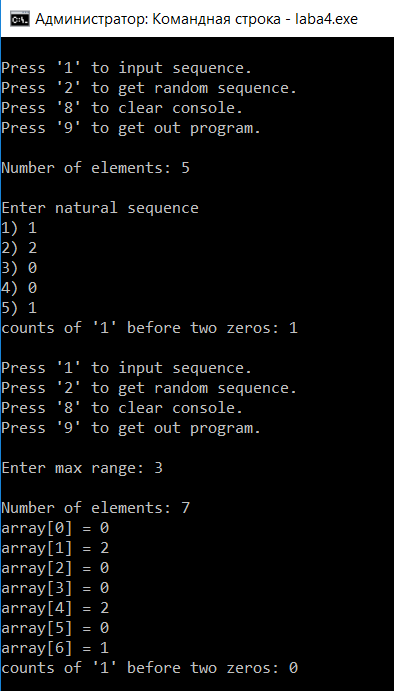
}

return main();

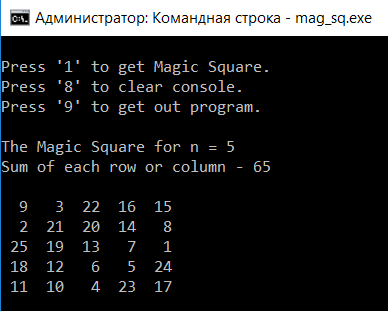
}

# ДОДАТОК В (Результат роботи програми)

## Перша програма



## Друга програма



# ДОДАТОК Г (Блок-схема)

