МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут комп'ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

ЗВІТ

До лабораторної роботи № 2

На тему: "*Документування етапів проектування та кодування програми* **"**

З дисципліни: *"Вступ до інженерії програмного забезпечення"*

Лектор:
Левус Є.В.
Виконав:
ст. гр. ПЗ-15
Процiв О.М.

Прийняв:
Самбір А.А.

« ____ » _____ 2022 р.

∑= ____ .....

Львів – 2022

Тема

Документування етапів проектування та кодування програми.

Мета

Навчитися документувати основні результати етапів проектування та кодування найпростіших програм.

32, 20, 7

Теоретичні відомості

7. У моїй програмі використовується однозв’язний список та масив.

Зв’язний список - структура даних, в якій елементи лінійно впорядковані, але порядок визначається не номерами елементів, а вказівниками, які входять в склад елементів списку та вказують на наступний за елемент

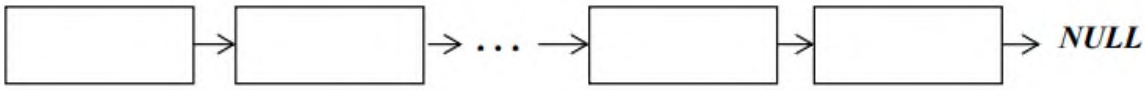


Рис. 4. Графічне зображення однозв’язного списку

Масив - об’єднання однотипних елементів

Масиви повинні мати:

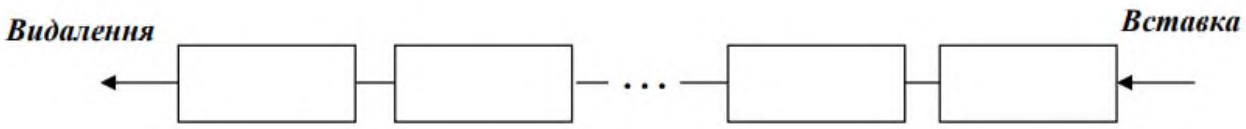
- елементи одного типу;
- фіксований набір елементів;
- кожний елемент має унікальний індекс (або набір індексів у випадку багатовимірного масиву);
- кількість індексів визначають розмірність масиву;
- звернення до елемента масиву виконується за ім’ям масиву і значенням індексів для даного елемента;
- у пам’яті елементи масиву розміщуються підряд.

Альтернативні структури даних:

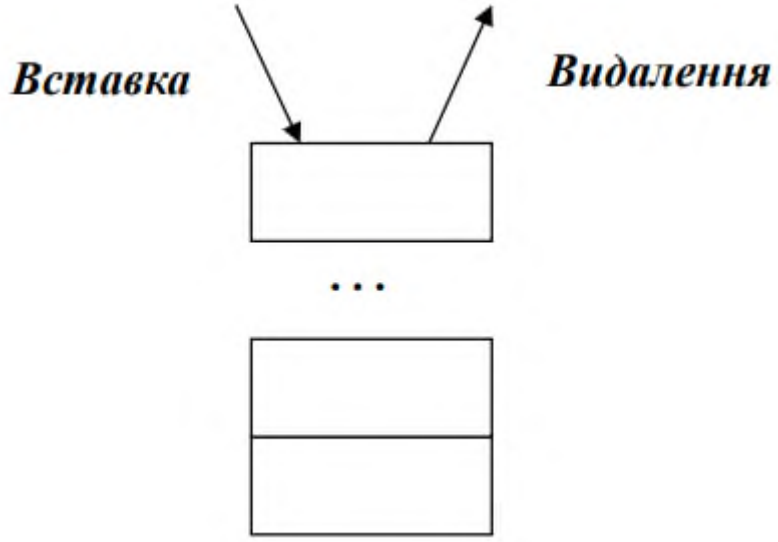
Структури - об’єднання різнотипних елементів (поля структури)

Перелічувані типи - тип даних, що описується шляхом перелічення всіх можливих значень (кожне з яких позначається власним ідентифікатором), які можуть приймати об’єкти даного типу. Тобто перелічуваний тип – це набір ідентифікаторів, які відіграють ту ж роль, що і звичайні іменовані константи, але пов’язані з одним типом.

Черга - динамічна структура даних, що працює за принципом для опрацювання елементів «перший прийшов – перший пішов». У черги є голова та хвіст. Елемент, що додається до черги, опиняється в її хвості



Стек – динамічна структура даних, що працює за принципом «перший прийшов – останній пішов». Усі операції (наприклад, видалення, вставка елементу) зі стеком можна проводити тільки з одним елементом, який знаходиться на «верхівці» стеку та був доданий останнім.



Клас - абстрактний тип даних, що визначається користувачем і є моделлю реального об’єкта у вигляді даних та функцій для роботи з ними

20. Форматування умов if-else:

(кожна команда починається після відступу у 2 пробіли, else пишеться в тому ж рядку, що і закриваюча дужка)

Форматування умов switch:

(кожна команда починається після відступу у 4 пробіли, а case – після відступу у 2 пробіли)

Форматування циклів:

(кожна команда починається після відступу у два пробіли, відкриваюча дужка пишеться у рядку заголовку циклу)

32. Рефакторинг коду – один з типових процесів, що полягає у перетворенні програмного коду, зміні внутрішньої структури програмного забезпечення для полегшення розуміння коду і легшого внесення подальших змін без зміни зовнішньої поведінки самої системи

Постановка завдання

Частина 1.

У розробленій раніше програмі до лабораторної роботи з дисципліни «Основи програмування» внести зміни – привести її до модульної структури, де модуль – окрема функція-підпрограма. У якості таких функцій запрограмувати алгоритми зчитування та запису у файл, сортування, пошуку, редагування, видалення елементів та решта функцій згідно варіанту.

Частина 2.

Сформувати пакет документів до розробленої раніше власної програми:

1. схематичне зображення структур даних, які використовуються для збереження інформації ;

2. блок-схема алгоритмів – основної функції й двох окремих функцій-підпрограм (наприклад, сортування та редагування);

3. текст програми з коментарями та оформлений згідно вище наведених рекомендацій щодо забезпечення читабельності й зрозумілості. Для схематичного зображення структур даних, блок-схеми алгоритму можна використати редактор MS-Visio або інший редактор інженерної та ділової графіки.

Виконання роботи

2.1

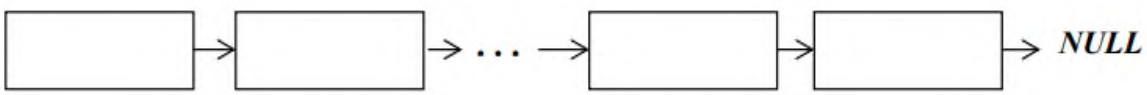
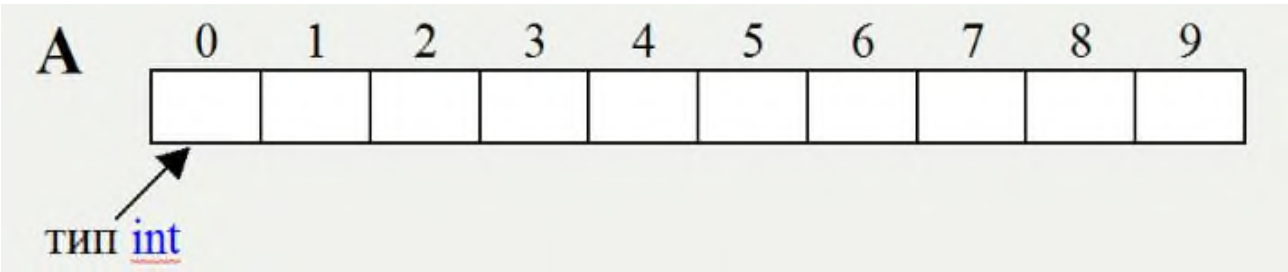
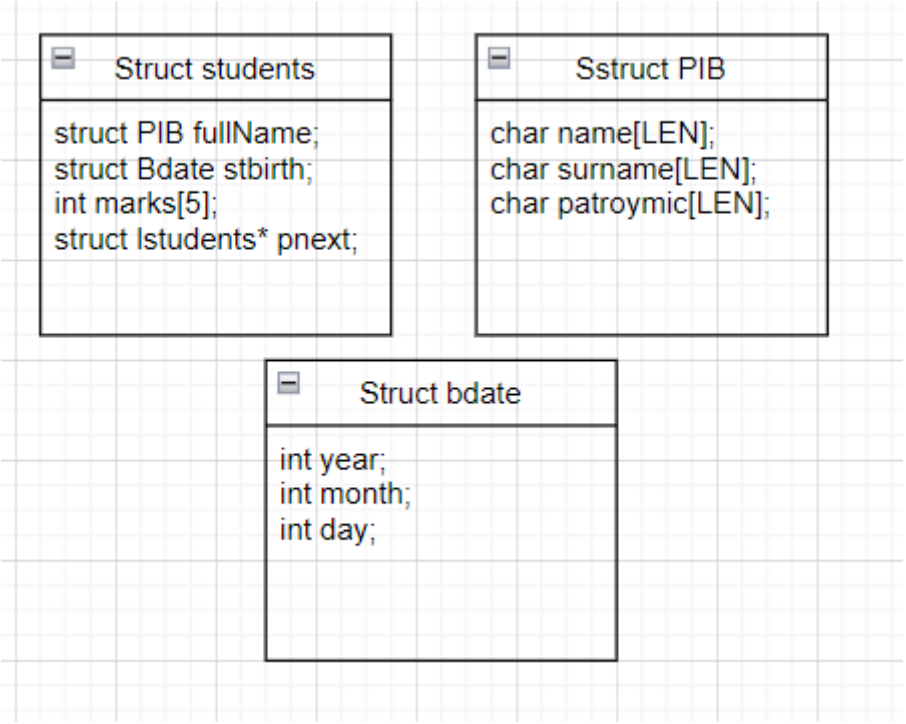


Рис. 4. Графічне зображення однозв'язного списку

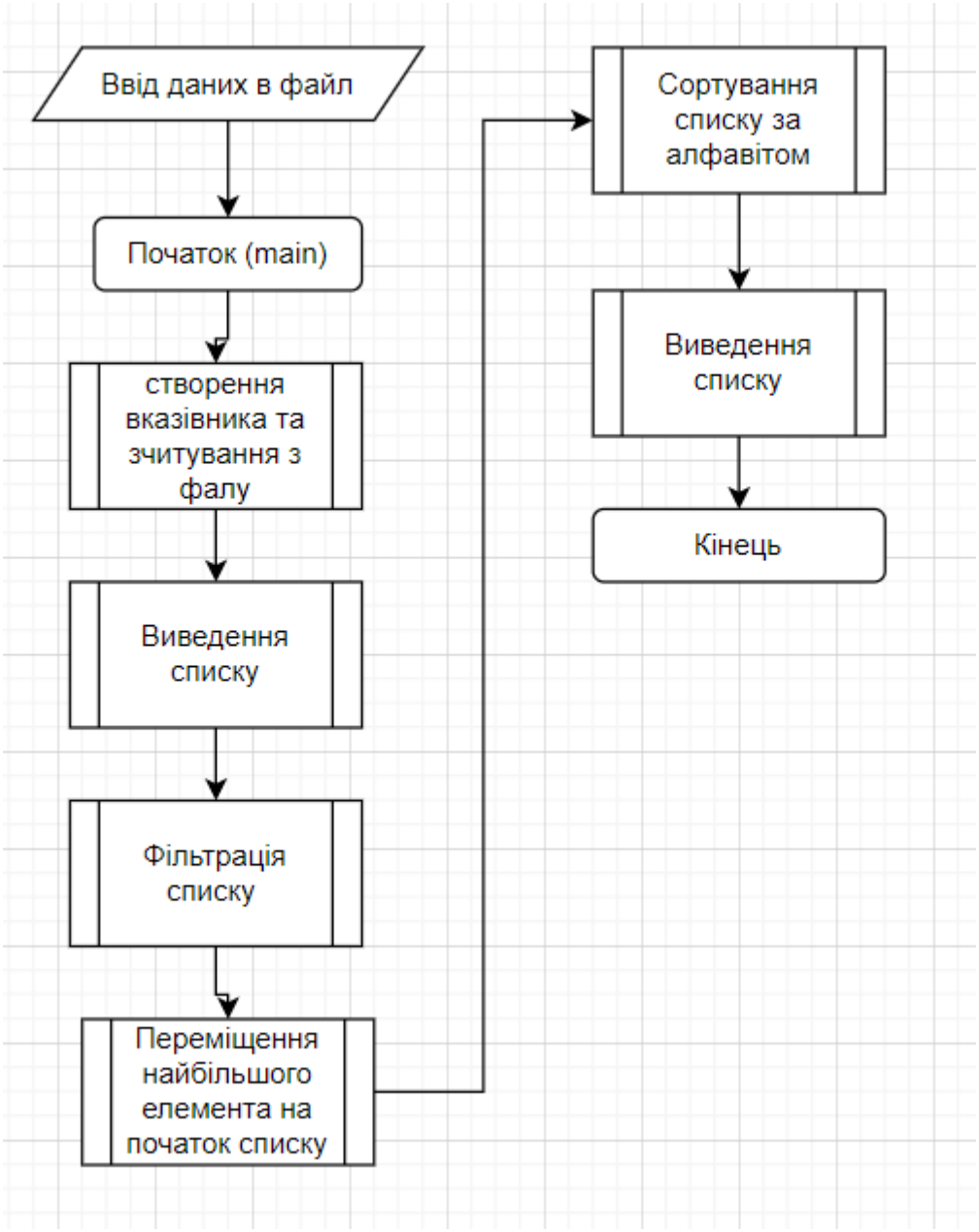
Графічне зображення масиву



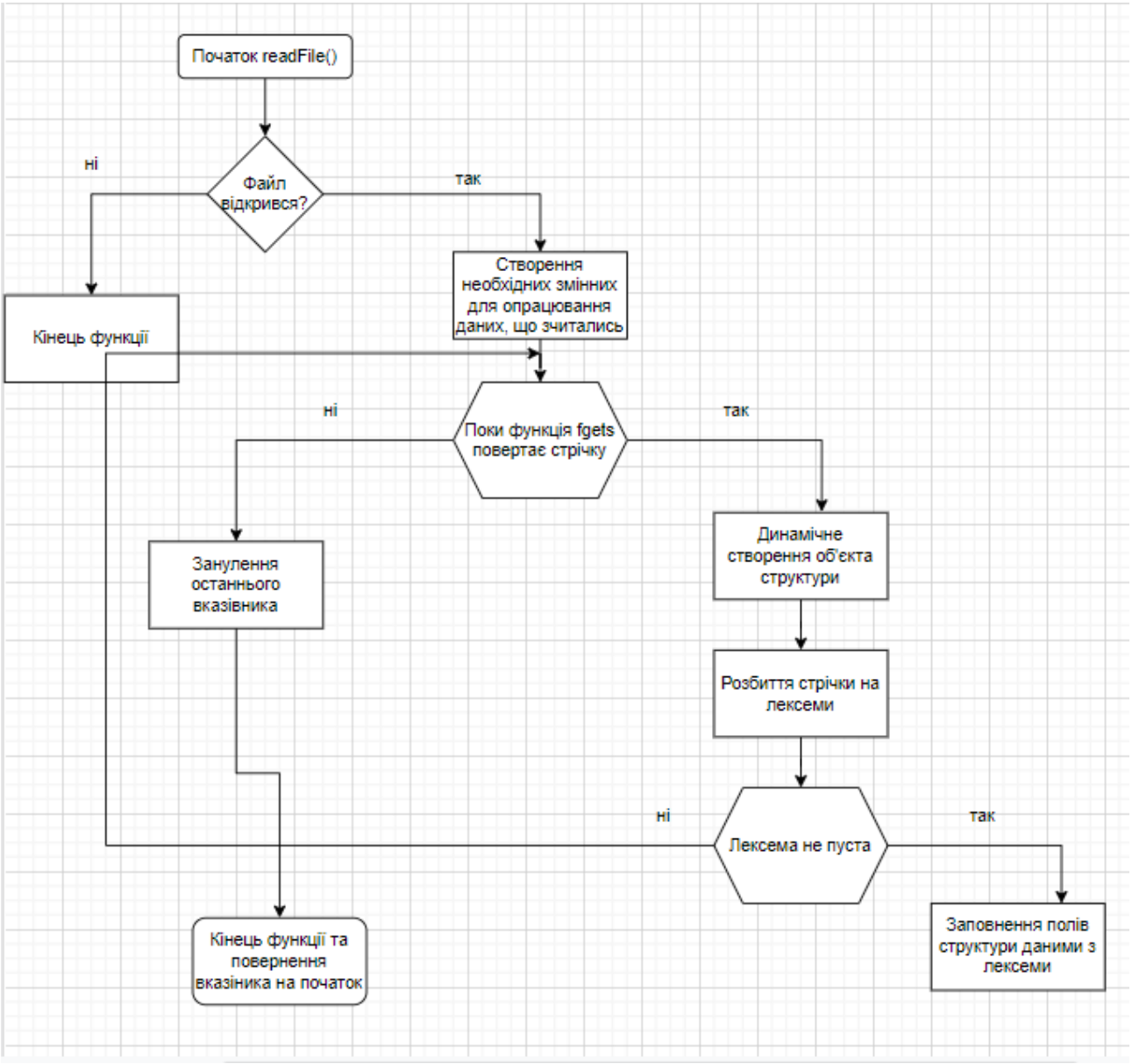


2.2 Блок-схеми

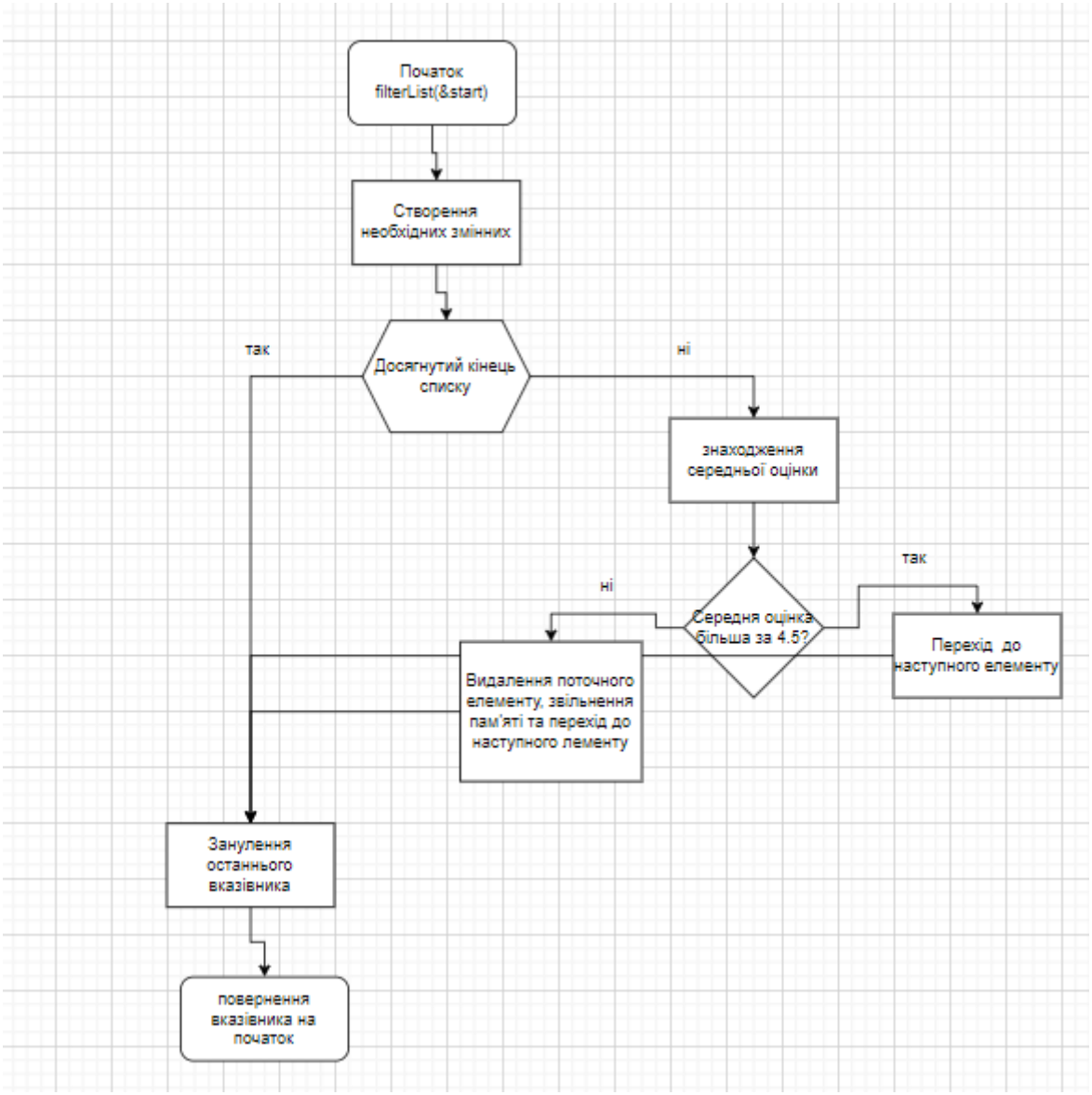
Основна функція



Функція читання з фалу:



Функція фільтрації списку:



2.3 Код програми

Functions.h

```
#define _CRT_SECURE_NO_WARNINGS
#ifndef functions
#define functions
#define LEN 250
#define NSTUDENTS 10
#define NMARKS 5
// допоміжна структура з даними студента
struct PIB
{
	char name[LEN];
	char surname[LEN];
	char patroymic[LEN];
};
// допоміжна структура з датою народження студента
struct Bdate
{
	int year;
	int month;
	int day;
};
// основний однозв'язний список
typedef struct lstudents {
	struct PIB fullName;
	struct Bdate stbirth;
```

```
	int marks[5];
	struct lstudents* pnext;
} students;
// функція зчитуванню даних з файлу
students* readFile();
// функція фільтрації списку
students* filterList(students** start);
// функція друкування списку
void printList(students** start);
//ддопоміжна функція по знаходженню найвизого елемента та переміщення його на початок
void findhighest(students** p);
//функція сортування списку
void sortlist(students** p);

#endif
```

filterList.c

```
#include "functions.h"
#include <stdio.h>

students* filterList(students** start) {
	students* pworkwith = *start;
	students* p = *start;
	double sum = 0, avg = 0;
	while (p->pnext != NULL) {
		for (int i = 0; i < NMARKS; i++) {
			sum += p->marks[i];
		}
		avg = sum / NMARKS;
		sum = 0;

		if (avg <= 4.5000001) {
			pworkwith->pnext = p->pnext;
			free(p);
			p = pworkwith;
		}
		pworkwith = p;
		p = p->pnext;
		if (p->pnext == NULL) {
			pworkwith->pnext = p->pnext;
			free(p);
			p = pworkwith;
		}
	}
	p->pnext = NULL;
	return *start;
}
```

findHighest.c

```
#include "functions.h"
#include <stdio.h>

void findhighest(students** p) {
	students* pwork = *p;
	students* pstart = *p;
	students* pprev = NULL;
	students* phigh = pwork;
	students* ptemp = NULL;
	students* ptest = NULL;
	while (pwork != NULL) {
		if (pwork->fullName.surname[0] > phigh->fullName.surname[0]) {
			pprev = ptest;
			phigh = pwork;
		}
		ptest = pwork;
		pwork = pwork->pnext;
	}
	if (phigh != pstart) {
```

```
                pprev->pnext = pstart;
                ptemp = pstart->pnext;
                pstart->pnext = phigh->pnext;
                phigh->pnext = ptemp;
                pstart = phigh;
        }

        *p = pstart;

}
```

printList.h

```
#define _CRT_SECURE_NO_WARNINGS
#include "functions.h"

#include <string.h>
#include <stdlib.h>



void printList(students** start) {
        students* pworkwith = *start;
        while (pworkwith != NULL) {
                printf("%15s\t%s\t%s\t", pworkwith->fullName.name, pworkwith-
>fullName.surname, pworkwith->fullName.patroymic);
                printf("%d,%d,%d\t", pworkwith->stbirth.day, pworkwith->stbirth.month,
pworkwith->stbirth.year);
                for (int i = 0; i < NMARKS; i++) {
                        printf("%d ", pworkwith->marks[i]);
                }
                printf("\n");
                pworkwith = pworkwith->pnext;
        }
}
```

readFile.c

```
#include "functions.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

students* readFile() {
        FILE* fstudents = fopen("students.txt", "r");
        if (fstudents == NULL) {
                printf("Error with opening the file");
                return 1;
        }

        char fromfile[LEN];
        char* leks;
        int countField = 1;
        int moveinarr = 0;
        int numFromleks = 0;
        int countmarks = 0;
        students* pstart = NULL;
        students* p = (students*)malloc(sizeof(students));

        while (fgets(fromfile, LEN, fstudents)) {
                students* anp = (students*)malloc(sizeof(students));
                p->pnext = anp;
                p = anp;
                if (pstart == NULL) {
                        pstart = p;
                }

                leks = strtok(fromfile, " , . ");
                while (leks != NULL) {
```

```
                        if (countField <= 3) {
                                if (countField == 1) { strcpy(p->fullName.name, leks);
countField++; }
                                else if (countField == 2) { strcpy(p->fullName.surname, leks);
countField++; }
                                else { strcpy(p->fullName.patroymic, leks); countField++; }
                        }
                        else if (countField > 3 && countField <= 6) {
                                numFromleks = atoi(leks);
                                if (countField == 4) { p->stbirth.day = numFromleks;
countField++; }
                                else if (countField == 5) { p->stbirth.month = numFromleks;
countField++; }
                                else { p->stbirth.year = numFromleks; countField++; }

                        }
                        else {
                                numFromleks = atoi(leks);
                                p->marks[countmarks] = numFromleks;
                                countmarks++;
                        }

                        leks = strtok(NULL, " , .  ");

                }
                countmarks = 0;
                countField = 1;
                moveinarr++;
        }
        p->pnext = NULL;
        return pstart;
}
```

sortList.c

```
#include "functions.h"
#include <stdio.h>

void sortlist(students** p) {
        students* pwork = *p;
        students* pstart = *p;
        students* ptemp = NULL;
        students* pprev = NULL;
        students* curelem = pwork;
        while (pwork != NULL) {
                curelem = pstart;
                while (curelem->pnext != NULL) {
                        if (curelem->fullName.surname[0] < curelem->pnext->fullName.surname[0])
{
                                ptemp = curelem->pnext;
                                curelem->pnext = ptemp->pnext;
                                pprev->pnext = ptemp;
                                ptemp->pnext = curelem;


                        }
                        pprev = curelem;
                        curelem = curelem->pnext;
                        if (curelem == NULL) break;
                }
                pwork = pwork->pnext;
        }
        *p = pstart;
}
```

Main.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "functions.h"
```

```
int main()
{

	students* start = readFile();
	students* pworkwith;
	pworkwith = start;
	printList(&start);

	pworkwith = filterList(&start);

	findhighest(&start);
	printf("\n");
	sortlist(&start);
	printf("\n\n");
	printList(&start);

	return 0;
}
```

Висновок: виконавши лабораторну роботу, я навчився документувати етапи проектування та кодування програми. Також я розробив блок-схеми основних алгоритмів та здійснив рефакторинг коду.