МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ «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:
доц. кафедри ПЗ
Левус Є. В.

Виконав:
ст. гр. ПЗ-16
Чаус О. М.

Прийняв:
викладач кафедри ПЗ Самбір А. А.

« ____ » _____ 2022 р.

∑= _____ .....:

Львів-2022

Тема роботи: Документування етапів проектування та кодування програми.

Мета роботи: Навчитися документувати основні результати етапів проектування та кодування найпростіших програм.

Теоретичні відомості

33. Що таке статичний аналіз коду?

Статичний аналіз коду – це метод відлагодження програми шляхом аналізу її коду до запуску використовуючи певні інструменти.

19. Як зробити текст програми читабельним?

Щоб зробити текст програми читабельним, необхідно правильно форматувати код, зокрема:

- використовувати правильну табуляцію
- не перевищувати стандарту довжину рядків(80 символів)
- росподіляти функції та методи коментарями довжиною 80 символів
- називати функції та змінні відповідно до конвенцій про імена(угорська нотація і т. д.)

6. Що таке шаблони проектування? Яка мета використання патернів проектування?

Шаблони проектування – це архітектурна конструкція, яка надає рішення проблеми проектування в певному контексті. Це – опис того, як можна вирішити завдання. Патери проектування використовуються для вирішення поширених проблем, пришвидшуючи процес розробки.

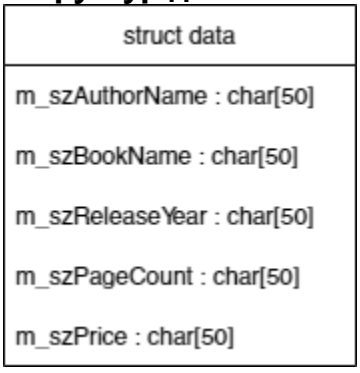
Постановка завдання

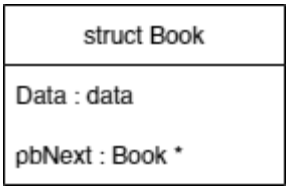
Частина І. У розробленій раніше програмі до лабораторної роботи з дисципліни «Основи програмування» внести зміни – привести її до модульної структури, де модуль – окрема функція-підпрограма. У якості таких функцій запрограмувати алгоритми зчитування та запису у файл, сортування, пошуку, редагування,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

Частина ІІ. 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 функційпідпрограм (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.

Отримані результати

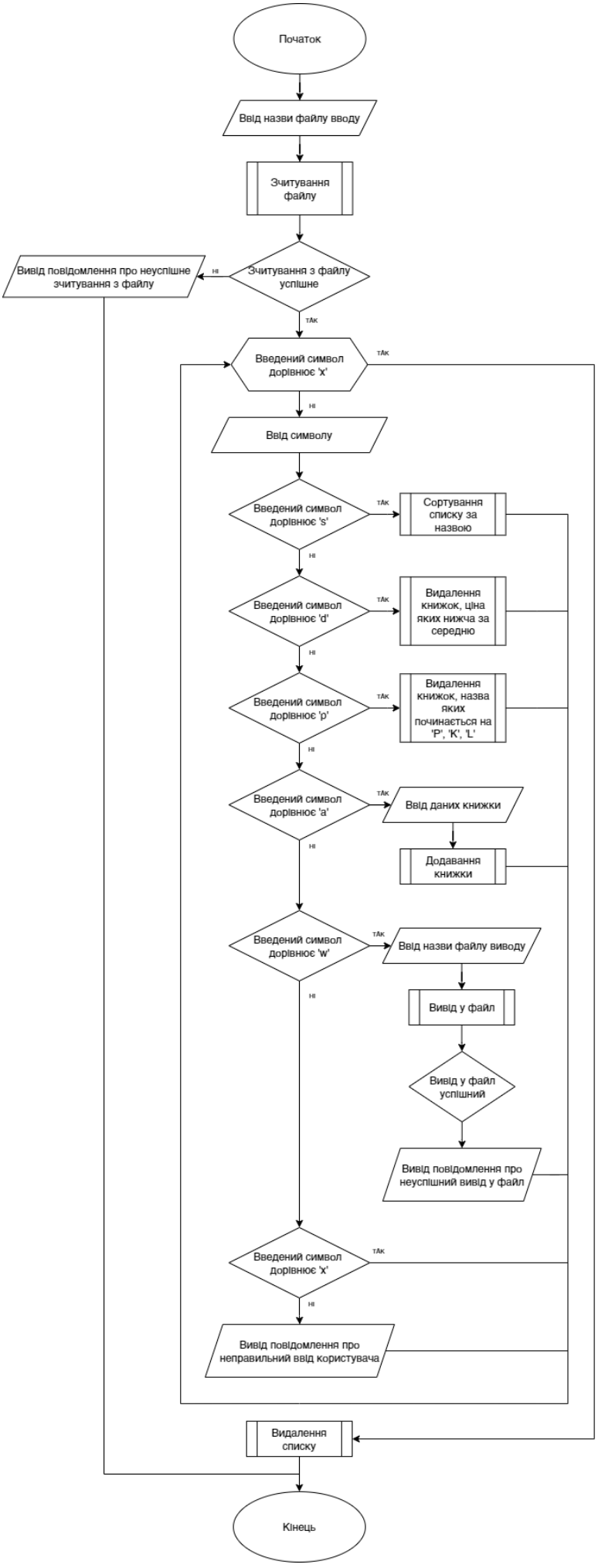
1. **Схематичні зображення структур даних, які використовуються**



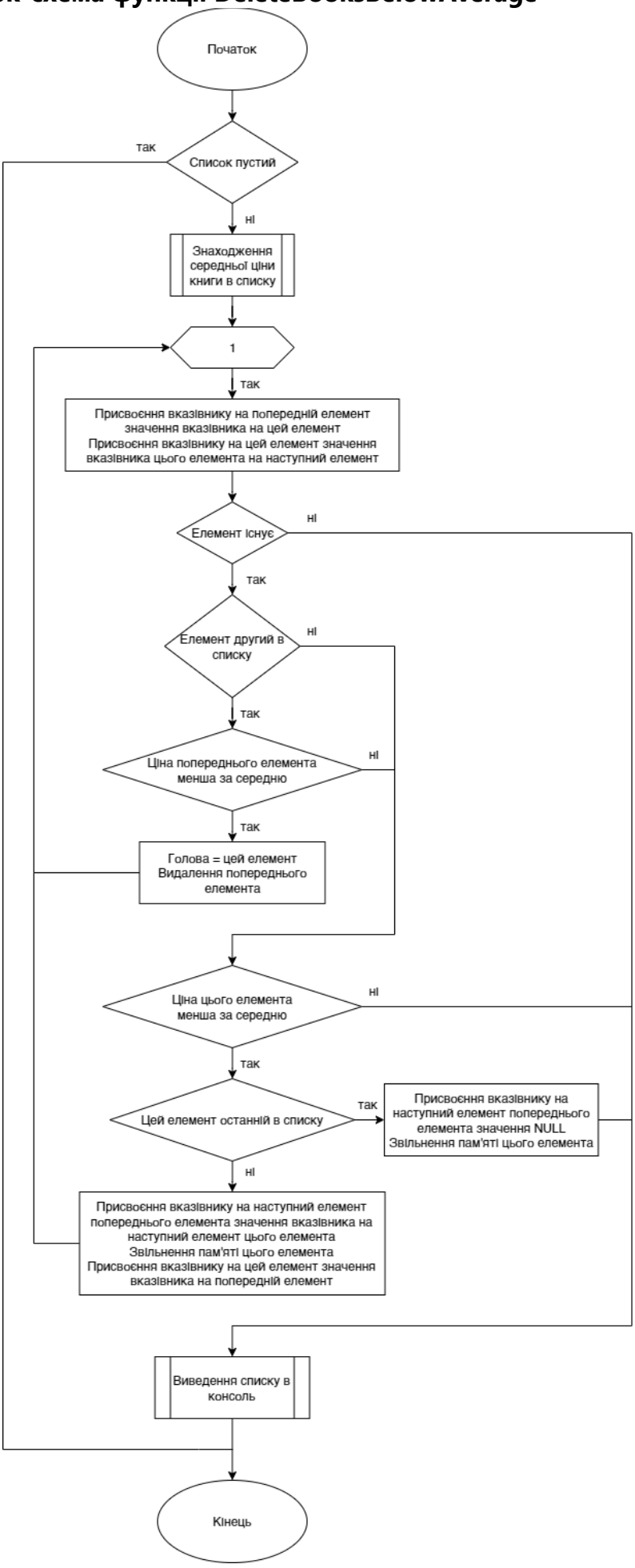


2. Блок-схема основної функції і двох окремих функцій-підпрограм

2.1. Блок-схема функції main


Початок
Ввід назви файлу вводу
Зчитування файлу
Зчитування з файлу успішне
ні
Вивід повідомлення про неуспішне зчитування з файлу
так
Введений символ дорівнює 'x'
так
ні
Ввід символу
Введений символ дорівнює 's'
так
Сортування списку за назвою
ні
Введений символ дорівнює 'd'
так
Видалення книжок, ціна яких нижча за середню
ні
Введений символ дорівнює 'p'
так
Видалення книжок, назва яких починається на 'P', 'K', 'L'
ні
Введений символ дорівнює 'a'
так
Ввід даних книжки
Додавання книжки
ні
Введений символ дорівнює 'w'
так
Ввід назви файлу виводу
Вивід у файл
Вивід у файл успішний
Вивід повідомлення про неуспішний вивід у файл
ні
Введений символ дорівнює 'x'
так
ні
Вивід повідомлення про неправильний ввід користувача
Видалення списку
Кінець

2.2. Блок-схема функції DeleteBooksBelowAverage



2.3 Блок-схема функції FindAveragePrice

2.3. Текст програми з коментарями та оформлений згідно рекомендацій

Файл functions.h:

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

#include <stdlib.h>
#include <stdio.h>
//default size of string used in struct data
#define STR_SIZE 50
//default size of line used for read from file
#define LINE_SIZE 150

typedef struct
{
	char	m_szAuthorName[STR_SIZE];
	char	m_szBookName[STR_SIZE];
	char	m_szReleaseYear[STR_SIZE];
	char	m_szPageCount[STR_SIZE];
	char	m_szPrice[STR_SIZE];
```

```
} data;
typedef struct
{
	data   Data;
	struct Book *pbNext;
} Book;
//writes to file, returns 1 if successful, returns 0 if not successful
int WriteToFile(char * szFileName, Book* head);
//reads from file, takes null pointer, returns 1 if successful, returns 0
//
	if not successful
int ReadFromFile(char *szFileName, Book** head);
//prints out whole list
void PrintList(Book *head);
//returns average price of elements in the list
double FindAveragePrice(Book *head);
//deletes all books priced below average, takes pointer to head pointer
void DeleteBooksBelowAverage(Book **head);
//sorts list by name of the book in descending order, takes pointer to head pointer
void SortByName(Book **head);
//deletes every book containing "PKL" at the start of it's name, takes pointer
//
		to head pointer
void DeleteBooksStartingWithWithPKL(Book **head);
//adds new book placed by name in descending order, takes pointer to head pointer
void AddBook(Book **head);
//deletes list, takes pointer to head pointer
void DeleteList(Book** head);
#endif
```

Файл functions.c:

```
#pragma warning(disable:4996)
#include "functions.h"

//-----------------------------------------------------------------------------
int WriteToFile(char *szFileName, Book *head)
{
	FILE *pfOutput = fopen(szFileName, "w");
	if (head)
	{
		Book* pbCurrentItem = head;
		while (pbCurrentItem)
		{
			fprintf(pfOutput, "%s\t%s\t%s\t%s\t%s",
				pbCurrentItem->Data.m_szAuthorName,
				pbCurrentItem->Data.m_szBookName,
				pbCurrentItem->Data.m_szReleaseYear,
				pbCurrentItem->Data.m_szPageCount,
				pbCurrentItem->Data.m_szPrice);
			pbCurrentItem = pbCurrentItem->pbNext;
			if (pbCurrentItem)
				fprintf(pfOutput, "\n");
		}
		return 1;
	}
	fclose(pfOutput);
	return 0;
}
//-----------------------------------------------------------------------------
int ReadFromFile(char *szFileName, Book **head)
{
	FILE* pfInput;
	if (pfInput = fopen(szFileName, "r"))
	{
		char* szLim = ",\t\n";
		char cTemp = fgetc(pfInput);
		if(!cTemp)
		{
```

```
                return 0;
        }
        else { ungetc(cTemp, pfInput); }
        Book* pbCurrentItem = (Book*)malloc(sizeof(Book));
        char line[LINE_SIZE];
        while (1)
        {
                if (!*head)
                        *head = pbCurrentItem;
                fgets(line, sizeof(line), pfInput);
                strcpy(pbCurrentItem->Data.m_szAuthorName, strtok(line, szLim));
                strcpy(pbCurrentItem->Data.m_szBookName, strtok(NULL, szLim));
                strcpy(pbCurrentItem->Data.m_szReleaseYear, strtok(NULL, szLim));
                strcpy(pbCurrentItem->Data.m_szPageCount, strtok(NULL, szLim));
                strcpy(pbCurrentItem->Data.m_szPrice, strtok(NULL, szLim));
                if (feof(pfInput))
                {
                        pbCurrentItem->pbNext = NULL;
                        break;
                } else
                {
                        Book* pbNextItem = (Book*)malloc(sizeof(Book));
                        pbCurrentItem->pbNext = pbNextItem;
                        pbCurrentItem = pbNextItem;
                }
        }
        fclose(pfInput);
        printf("\t\t\t\t\t\t\t[INITIAL LIST]\n\n");
        PrintList(*(head));
        return 1;
    }
    return 0;
}
//---------------------------------------------------------------------------
void PrintList(Book *head)
{
    putchar('\n');
    for (int i = 0; i < 123; i++)
    {
        if (i == 0 || i == 122 || i == 33 || i == 73 || i == 89 || i == 105)
        {
            putchar('+');
        } else putchar('-');
    }
    putchar('\n');
    printf("| \tAUTHOR NAME\t\t | \tBOOK NAME\t\t\t | \tYEAR\t |\tPAGES\t"
        " | \tPRICE\t   | \n");
    for (int i = 0; i < 123; i++)
    {
        if (i == 0 || i == 122 || i == 33 || i == 73 || i == 89 || i == 105)
        {
            putchar('+');
        } else putchar('-');
    }
    putchar('\n');
    while (head)
    {
        printf("| \t%-20s\t | \t%-24s\t | \t%-4s\t | \t%-4s\t | \t%-4s\t   | \n",
            head->Data.m_szAuthorName,
            head->Data.m_szBookName,
            head->Data.m_szReleaseYear,
            head->Data.m_szPageCount,
            head->Data.m_szPrice);
        head = head->pbNext;
    }
    for (int i = 0; i < 123; i++)
    {
        if (i == 0 || i == 122 || i == 33 || i == 73 || i == 89 || i == 105)
```

```
		{
			putchar('+');
		} else putchar('-');
	}
	printf("\n\n");
}
//-------------------------------------------------------------------------
double FindAveragePrice(Book *head)
{
	double dSum = 0;
	int i = 0;
	Book* pbCurrentItem = head;
	while (pbCurrentItem)
	{
		dSum += atof(head->Data.m_szPrice);
		i++;
		pbCurrentItem = pbCurrentItem->pbNext;
	}
	return dSum / i;
}
//-------------------------------------------------------------------------
void DeleteBooksBelowAverage(Book **head)
{
	if (head)
	{
		double avg = FindAveragePrice(*head);
		Book* pbCurrentItem = *(head);
		while (1)
		{
			Book* pbPreviousItem = pbCurrentItem;
			pbCurrentItem = pbCurrentItem->pbNext;
			if (!pbCurrentItem) break;
			if (pbPreviousItem == *(head) &&
				atof(pbPreviousItem->Data.m_szPrice) < avg)
			{
				*head = pbPreviousItem->pbNext;
				free(pbPreviousItem);
			}
			else if (atof(pbCurrentItem->Data.m_szPrice) < avg)
			{
				if (!pbCurrentItem->pbNext)
				{
					pbPreviousItem->pbNext = NULL;
					free(pbCurrentItem);
					break;
				}
				pbPreviousItem->pbNext = pbCurrentItem->pbNext;
				pbCurrentItem->pbNext = NULL;
				free(pbCurrentItem);
				pbCurrentItem = pbPreviousItem;
			}
		}
		printf("\t\t\t\t\t\t\t[LIST AFTER DELETION]\n\n");
		PrintList(*(head));
	}
}
//-------------------------------------------------------------------------
void SortByName(Book **head)
{
	if (*head)
	{
		Book* pbIterator = *head;
		while (pbIterator)
		{
			Book* pbCurrentItem = *head;
			while (pbCurrentItem->pbNext)
			{
				Book* prevP = pbCurrentItem;
```

```
                        pbCurrentItem = pbCurrentItem->pbNext;
                        if (strcmp(
                                prevP->Data.m_szBookName,
                                pbCurrentItem->Data.m_szBookName) > 0)
                        {
                                data temp = prevP->Data;
                                prevP->Data = pbCurrentItem->Data;
                                pbCurrentItem->Data = temp;
                        }
                }
                pbIterator = pbIterator->pbNext;
        }
        printf("\t\t\t\t\t\t\t[SORTED LIST]\n\n");
        PrintList(*head);
    }
}
//-------------------------------------------------------------------------
void DeleteBooksStartingWithWithPKL(Book **head)
{
    if (*head)
    {
        Book* pbCurrentItem = *(head);
        while (1)
        {
            Book* pbPreviousItem = pbCurrentItem;
            pbCurrentItem = pbCurrentItem->pbNext;
            if (!pbCurrentItem) break;
            char* szLimiter = "PKL";
            for (int i = 0; *(szLimiter + i) != 0; i++)
            {
                if (*(head) == pbPreviousItem)
                {
                    if (*(pbPreviousItem->Data.m_szBookName) == *(szLimiter +
i))
                    {
                        *(head) = pbPreviousItem->pbNext;
                        free(pbPreviousItem);
                        break;
                    }
                }
                if (*(pbCurrentItem->Data.m_szBookName) == *(szLimiter + i))
                {
                    if (!pbCurrentItem->pbNext)
                    {
                        free(pbCurrentItem);
                        pbPreviousItem->pbNext = NULL;
                        pbCurrentItem = pbPreviousItem;
                        break;
                    } else
                    {
                        pbPreviousItem->pbNext = pbCurrentItem->pbNext;
                        pbCurrentItem->pbNext = NULL;
                        free(pbCurrentItem);
                        pbCurrentItem = pbPreviousItem;
                        break;
                    }
                }
            }
        }
    }
    printf("\t\t\t\t\t\t\t[LIST AFTER DELETION]\n\n");
    PrintList(*(head));
}
//-------------------------------------------------------------------------
void AddBook(Book **head)
{
    char* szLim = ",\t\n";
    char szNewBookData[LINE_SIZE];
```

```
	char cTemp = 0;
	getchar(cTemp);
	fgets(szNewBookData, sizeof(szNewBookData), stdin);
	Book* pbNewBook = (Book*)malloc(sizeof(Book));
	pbNewBook->pbNext = NULL;
	strcpy(pbNewBook->Data.m_szAuthorName, strtok(szNewBookData, szLim));
	strcpy(pbNewBook->Data.m_szBookName, strtok(NULL, szLim));
	strcpy(pbNewBook->Data.m_szReleaseYear, strtok(NULL, szLim));
	strcpy(pbNewBook->Data.m_szPageCount, strtok(NULL, szLim));
	strcpy(pbNewBook->Data.m_szPrice, strtok(NULL, szLim));
	Book* pbCurrentItem = *(head);
	if (*head)
	{
		while (1)
		{
			Book* pbPreviousItem = pbCurrentItem;
			pbCurrentItem = pbCurrentItem->pbNext;
			if (pbPreviousItem == *(head) && (strcmp(
				pbNewBook->Data.m_szBookName,
				pbPreviousItem->Data.m_szBookName) < 0))
			{
				*(head) = pbNewBook;
				pbNewBook->pbNext = pbPreviousItem;
				break;
			}
			if (!pbCurrentItem)
			{
				pbPreviousItem->pbNext = pbNewBook;
				pbNewBook->pbNext = NULL;
				break;
			}
			if (strcmp(
				pbNewBook->Data.m_szBookName,
				pbCurrentItem->Data.m_szBookName) < 0)
			{
				pbPreviousItem->pbNext = pbNewBook;
				pbNewBook->pbNext = pbCurrentItem;
				break;
			}
		}
		printf("\t\t\t\t\t\t\t[LIST AFTER ADDING NODE]\n\n");
		PrintList(*(head));
	} else
	{
		*(head) = pbNewBook;
		pbNewBook->pbNext = NULL;
	}
}
void DeleteList(Book** head)
{
	if (*head)
	{
		Book* currI = *head;
		*head = NULL;
		while (currI)
		{
			Book* prevP = currI;
			currI = currI->pbNext;
			free(prevP);
		}
	}
}
//---------------------------------------------------------------------------
```

Файл main.c:

```
#pragma warning(disable:4996)
#include <stdlib.h>
#include <stdio.h>
```

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

int main()
{
	Book* head = NULL;
	char cAction = '\0';
	char szFileName[20];
	printf("Please enter input file name\n");
	scanf("%s", szFileName);
	if (ReadFromFile(szFileName, &head))
	{
		while (cAction != 'x')
		{
			printf("What do you want to do?\n"
				"\nSORT LIST\t\t\t\t\t\t-\t's'"
				"\nDELETE ALL BOOKS PRICED LOWER THAN AVERAGE\t\t-\t'd'"
				"\nDELETE ALL BOOKS WITH NAME STARTING WITH 'P', 'K', 'L'\t-
\t'p'"
				"\nADD A NEW BOOK TO THE LIST\t\t\t\t-\t'a'"
				"\nWRITE TO FILE\t\t\t\t\t\t-\t'w'"
				"\nQUIT PROGRAM\t\t\t\t\t\t-\t'x'\n");
			scanf(" %c", &cAction);
			switch (cAction)
			{
			case 's':
				SortByName(&head);
				break;
			case 'd':
				DeleteBooksBelowAverage(&head);
				break;
			case 'p':
				DeleteBooksStartingWithWithPKL(&head);
				break;
			case 'a':
				printf("Please enter following book info:\nAuthor name [TAB] Book
name [TAB]"
					" Year of release [TAB] Page count [TAB] Price\n");
				AddBook(&head);
				break;
			case 'w':
				printf("Please enter output file name:\n");
				scanf("%s", szFileName);
				if (!WriteToFile(szFileName, head))
				{
					printf("\n[WRONG FILENAME]\n\n");
				}
				getchar();
				break;
			case 'x':
				break;
			default:
				printf("\n[WRONG INPUT]\n\n");
				break;
			}
		}
		DeleteList(&head);
		return 0;
	}
	printf("Incorrect file name");
	return 0;
}
```

Висновок: під час виконання лабораторної роботи навчився формувати найпростіше ТЗ для розробки програмного забезпечення.