МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **ІКНІ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 2

На тему: *“ЕТАПИ ПРОЕКТУВАННЯ ТА КОДУВАННЯ ПРОГРАМИ”*

З дисципліни: *“Вступ до інженерії програмного забезпечення”*

Лектор:
доцент каф. ПЗ
Левус Є. В.

Виконав:
ст. гр. ПЗ-15
Шпортко Т. О.

Прийняв:
асистент каф. ПЗ
Самбір А. А.

« ____ » _____ 2023 р.

∑= ____ .....

Львів – 2023

Тема роботи: Етапи проектування та кодування програми.

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

ТЕОРЕТИЧНІ ВІДОМОСТІ

27) Як записуються класи та їх складові у мові С++?

У мові С++, класи є типом даних, який дозволяє створювати нові складені типи об'єктів. Класи можуть мати властивості (змінні) та методи (функції), які пов'язані з цими властивостями.

Для оголошення класу використовується ключове слово class, за яким слідує ім'я класу. Основні складові класу можуть бути визначені всередині тіла класу, що зазвичай знаходиться всередині файлу заголовка класу (з розширенням .h або .hpp).

Приклад опису класу «Студент» в С++:

```
class Cstudent {
  private:
    int nID;
    string sName;
    double dCourse;
    int nMarks[5];
  public:
    Student();
    Student(int nID, string sName, double dCourse, int nMarks[5]);
    void setStudent(int nID, string sName, double dCourse, int nMarks[5]);
    int getID();
    string getName();
    double getCourse();
    void print();
};
```

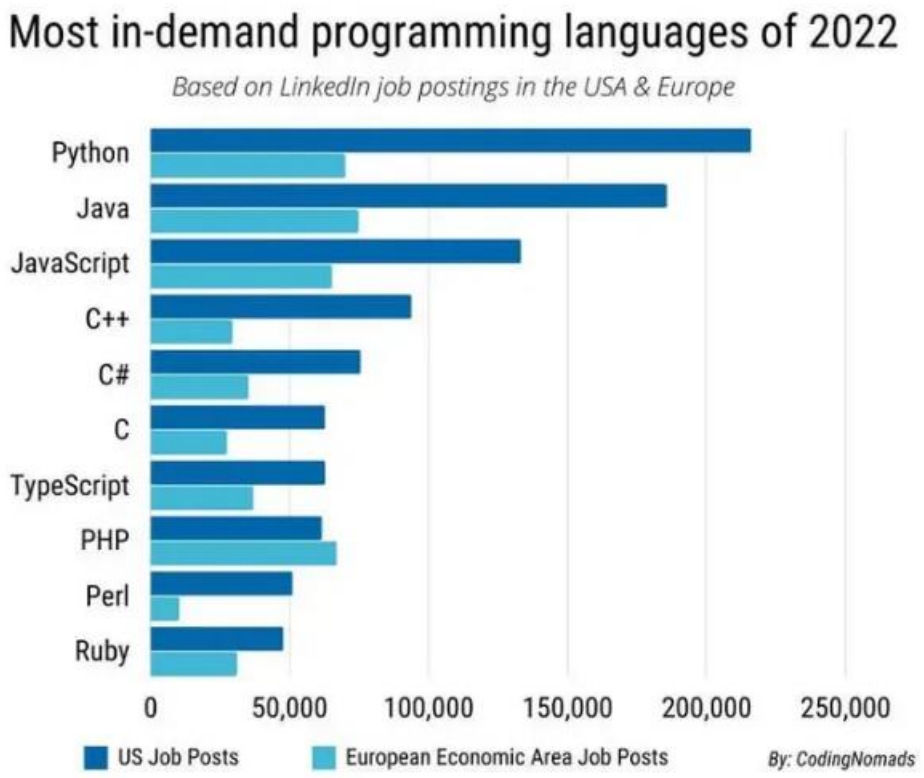
Цей клас містить у собі дані про ID студента, його ім’я, курс та 5 оцінок як змінні класу. Також клас містить конструктори, які проводять ініціалізацію для екземплярів класу та методи класу – функції, що дозволяють працювати з його об’єктами. Наприклад, метод getName повертає стрічку з ім’ям студента.

У класах можуть бути використані також модифікатори доступу - public, private та protected - для забезпечення контролю за тим, які властивості та методи можуть бути доступні ззовні класу або його нащадків. Крім того, можна

використовувати різні методи конструкторів та деструкторів для ініціалізації та очищення властивостей класу.

34) Перерахуйте найвикористовуваніші мови програмування.

Станом на 2022 рік маємо такі статистичні дані:



Python, Java, JavaScript, C++, C#, C, TypeScript, PHP, Perl, Ruby

2) У чому полягає етап проектування для найпростішої програми?

Типово проектування архітектури для найпростішої програми включає виконання таких завдань:

- трансформація вимог до програмного забезпечення в архітектуру (визначення структури програмної системи та її проектування);
- розбиття програмної системи на окремі компоненти та їх проектування з визначенням ключових елементів структур даних;
- розробка і документування інтерфейсів програми ;

Проектування полягає в створенні представлення про:

- складові програми;
- модульну структуру програмного продукту;
- алгоритмічну структуру;
- структури даних;

- вхідний і вихідний інтерфейс (вхідні і вихідні формати даних).

Для спрощення візуалізації процесу проектування використовуються так звані нотації – схематичне зображення характеристик розроблюваної програмної системи (блок-схеми, макети, діаграми і т.д.).

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 подібних функцій згідно варіанту.
Частина ІІ. Перевірити роботу програми згідно технічного завдання (п.3, п.4.1). У разі розбіжності внести корективи у програму.
Частина ІІІ. 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. схематичне зображення модульної структури програми у формі набору функцій;

4. текст програми з коментарями та оформлений згідно вище наведених рекомендацій щодо забезпечення читабельності й зрозумілості.

Для схематичного зображення структур даних, блок-схеми алгоритму, структури програми можна використати редактор MS-Visio або інший редактор інженерної та ділової графіки.

ХІД РОБОТИ

1. Схематичне зображення структур даних:

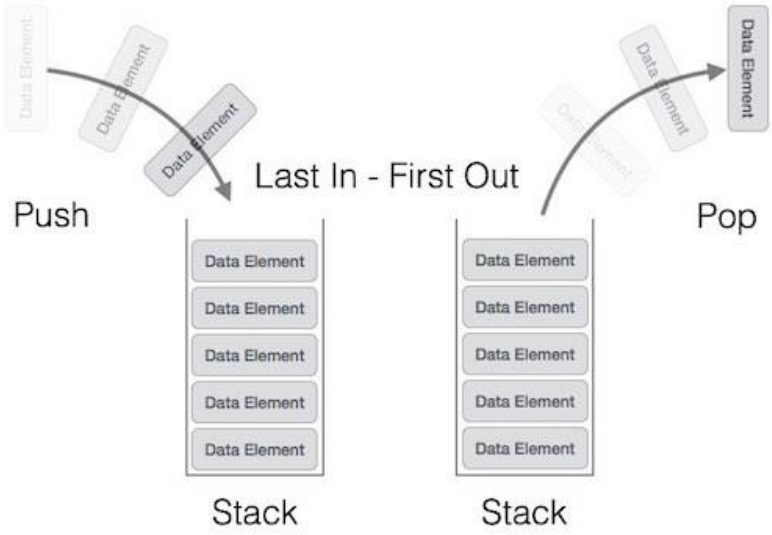


Рис 1. Стек (стос).

Рис 2. Файл.

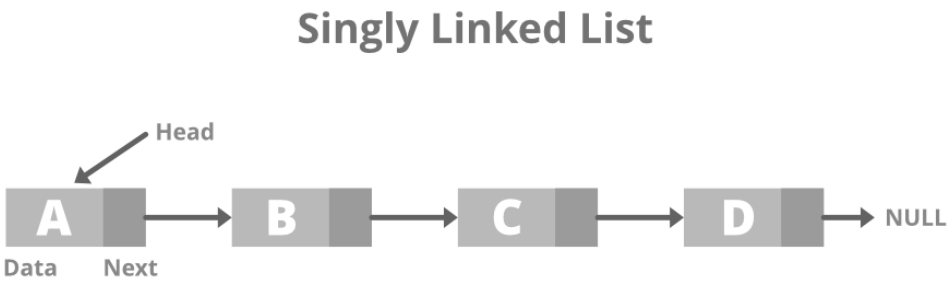


Рис 3. Однозв’язний список.

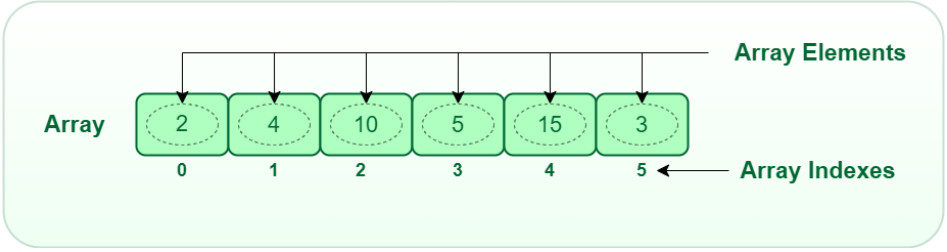


Рис 4. Масив.

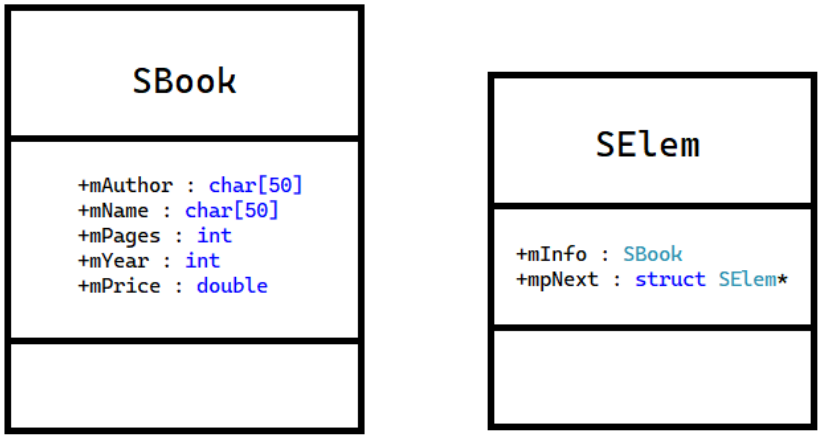


Рис 4. UML-діаграма структури.

2. Блок-схеми алгоритмів:

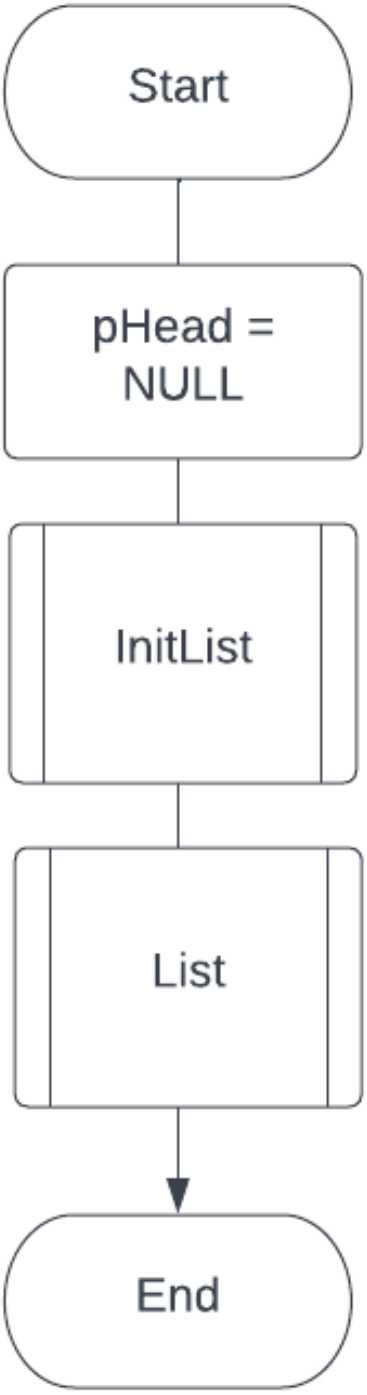


Рис. 5. Функція main.

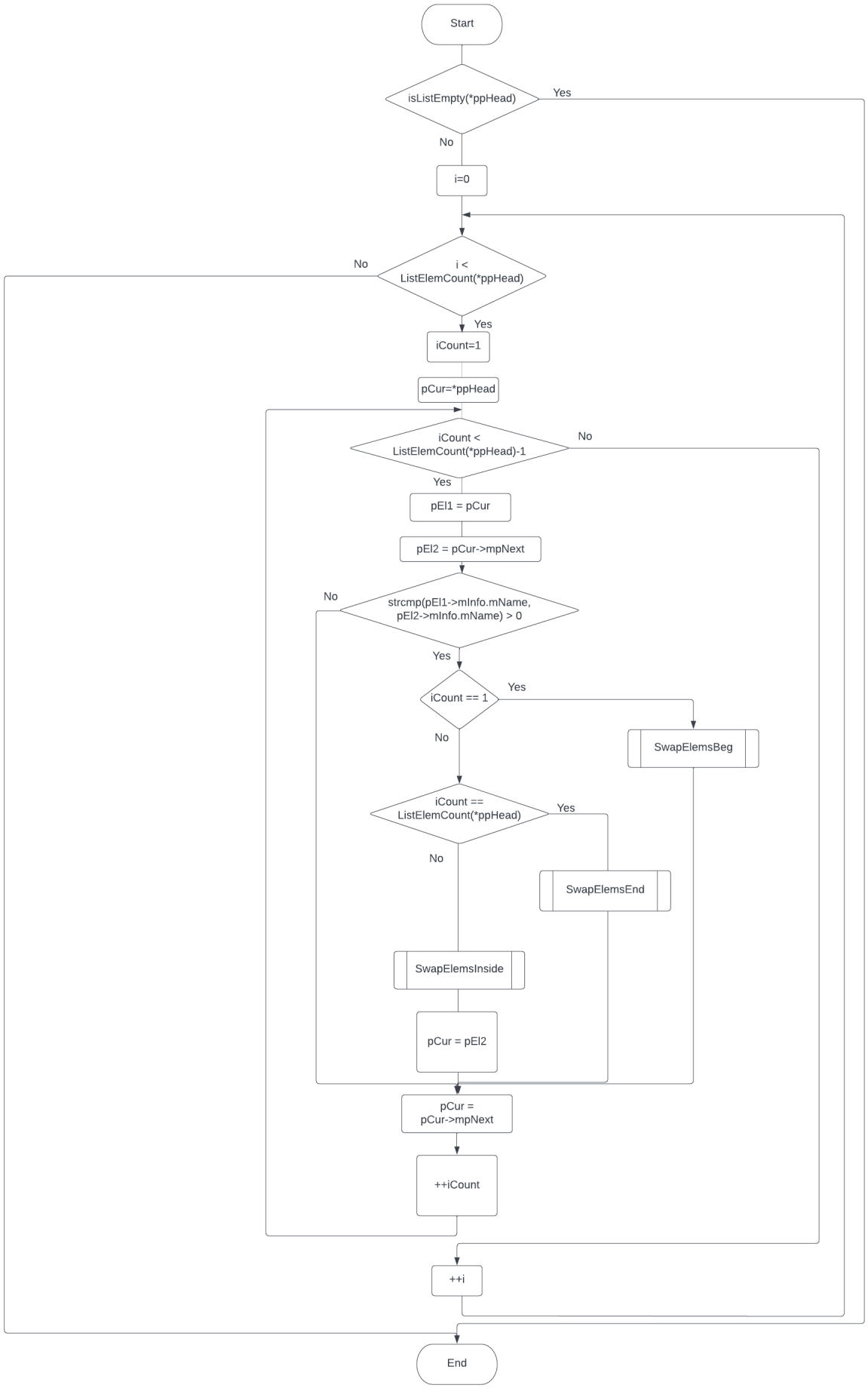


Рис. 6. Функція ListSortByName

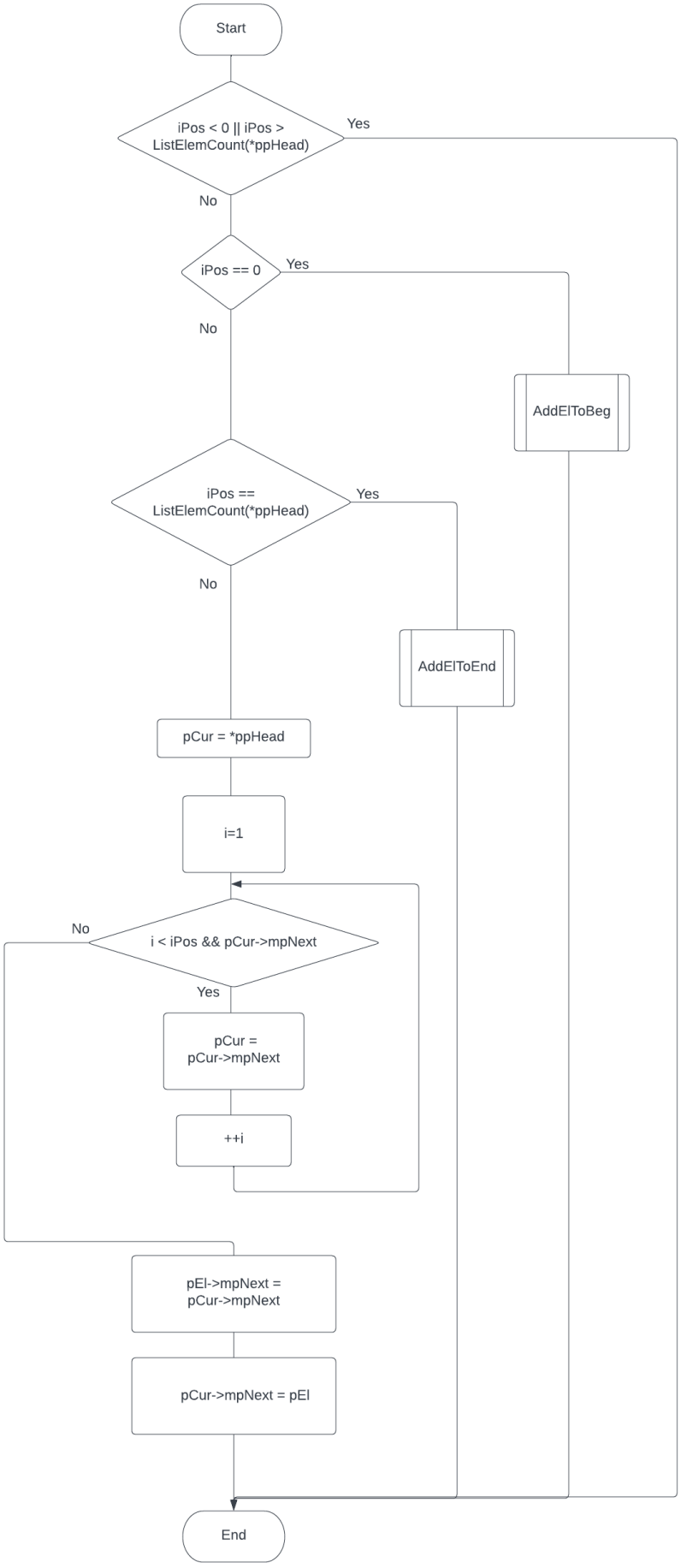


Рис. 7. Функція AddElToPos

3. Схематичне зображення модульної структури програми:

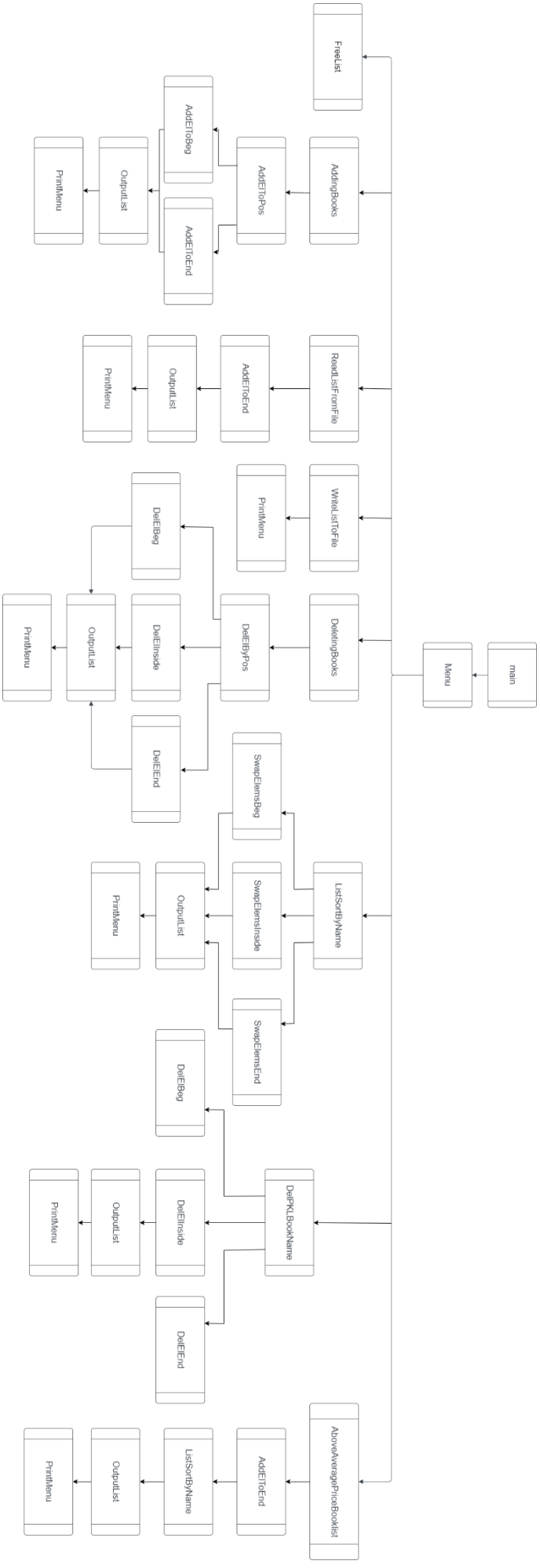


Рис. 8. Структура програми у вигляді набору функцій.

4. Текст програми:

input.txt

```
Taras Shevchenko;Kobzar;452;1840;22.61;
George Orwell;Nineteen Eighty-Four;328;1949;7.60;
Joanne K. Rowling;Harry Potter (Philosopher's Stone);223;1997;8.26;
Miguel de Cervantes;Don Quixote;1077;1605;13.99;
Plato;The Republic;568;-375;16.99;
Franz Kafka;The Metamorphosis;100;1915;4.49;
Virgil;The Aeneid;464;-19;12.56;
Andrzej Sapkowski;The Witcher:Time of Contempt;352;1995;9.99;
Valerian Pidmohylny;The City;288;1928;6.75;
Erich Maria Remarque;All Quiet on the Western Front;200;1929;10.5;
J. R. R. Tolkien;The Lord of the Rings;1178;1954;29.99;
Harper Lee;To Kill a Mockingbird;281;1960;7.99;
Gustave Flaubert;Madame Bovary;190;1856;7.99;
Victor Hugo;Les Miserables;1488;1862;14.99;
Marcel Proust;In Search of Lost Time;4215;1913;12.56
James Joyce;Ulysses;732;1922;8.76;
Charles Dickens;Bleak House;945;1852;8.49;
```

func.h

```
#ifndef FUNC_H
#define FUNC_H

#pragma warning(disable:4996)
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

typedef struct SBook {
	char mAuthor[50];
	char mName[50];
	int mPages;
	int mYear;
	double mPrice;
} SBook;

typedef struct SElem {
	SBook mInfo;
	struct SElem* mpNext;
} SElem;

void ReadListFromFile(SElem** ppHead, char* pchFileName);
void WriteListToFile(SElem* pHead, char* pchFileNameWP);
void OutputList(SElem* pHead);
void InitList(SElem** ppHead);
bool isListEmpty(SElem* pHead);
int ListElemCount(SElem* pHead);
void AddElToBeg(SElem** ppHead, SElem* pEl);
void AddElToPos(SElem** ppHead, SElem* pEl, int iPos);
void AddElToEnd(SElem** ppHead, SElem* pEl);
void AddBookToEnd(SElem** ppHead, char pchAuthor[50], char pchName[50], int iPages, int
iYear, double dbPrice);
void AddingBooks(SElem** ppHead);
void DelElBeg(SElem** ppHead);
void DelElInside(SElem* pHead, SElem* pDelete);
void DelElEnd(SElem* pHead);
void DelElByPos(SElem** ppHead, int iPos);
```

```
void DelPKLBookName(SElem** ppHead, int cCnt);
void DeletingBooks(SElem** ppHead);
void SwapElemsBeg(SElem** ppHead);
void SwapElemsInside(SElem* pHead, SElem* pEl);
void SwapElemsEnd(SElem* pHead);
void ListSortByName(SElem** ppHead);
int AboveAveragePriceBooklist(SElem* pHead, SElem** ppHead2);
void FreeList(SElem** ppHead);
void PrintStars(int iNum);
void PrintX(int iNum);
void PrintMenu();
void Menu(SElem** ppHead);

#endif
```

func.c

```
#include "func.h"

//-----------------------------------------------------------------------------

//This function reads data from text (.txt) file to a linked list. (needs filename)
void ReadListFromFile(SElem** ppHead, char* pchFileName) {

	FILE* fp;
	char* pchPtr;
	char szBuffer[100] = { 0 };

	fp = fopen(pchFileName, "r");
	if (fp == NULL) exit(1);


	while (!feof(fp)) {
		fgets(szBuffer, 100, fp);
		SElem* pEl = (SElem*)malloc(sizeof(SElem));

		if (pEl) {
			pchPtr = strtok(szBuffer, ";");
			strcpy(pEl->mInfo.mAuthor, pchPtr);
			pchPtr = strtok(NULL, ";");
			strcpy(pEl->mInfo.mName, pchPtr);
			pchPtr = strtok(NULL, ";");
			pEl->mInfo.mPages = atoi(pchPtr);
			pchPtr = strtok(NULL, ";");
			pEl->mInfo.mYear = atoi(pchPtr);
			pchPtr = strtok(NULL, ";");
			pEl->mInfo.mPrice = atof(pchPtr);
			AddElToEnd(ppHead, pEl);
		}
		else exit(4);
	}
	fclose(fp);

	printf("\n//List is read from \"input.txt//\"\n\n");
}

//-----------------------------------------------------------------------------

//This function writes data from linked list to text (.txt) or binary (.bin) file on
the user's choice. (needs filename)
void WriteListToFile(SElem* pHead, char* pchFileNameWP) {
	if (isListEmpty(pHead)) {
		printf("\nThe list is empty. Try reading it by typing \"1\" !");
		return;
	}

	int iMode;
```

```
	char szFileName[100] = { 0 };

	printf("\nType 0 if you want to write list in binary mode, otherwise it'll be
written in text mode.\n");
	scanf("%d", &iMode);

	if (iMode) {
		strncat(szFileName, pchFileNameWP, 50);
		strncat(szFileName, ".txt", 4);
		FILE* fp = fopen(szFileName, "w");
		if (fp == NULL) exit(1);

		for (SElem* pCur = pHead; pCur != NULL; pCur = pCur->mpNext) {
			fprintf(fp,	"%s;%s;%d;%d;%.2lf;\n",	pCur->mInfo.mAuthor,	pCur-
>mInfo.mName, pCur->mInfo.mPages, pCur->mInfo.mYear, pCur->mInfo.mPrice);
		}

		printf("\n//List is written to \"%s\"//\n\n", szFileName);
		fclose(fp);
	}
	else {
		strncat(szFileName, pchFileNameWP, 50);
		strncat(szFileName, ".bin", 4);
		FILE* fp = fopen(szFileName, "wb");
		if (fp == NULL) exit(1);

		char szBuffer[256] = { 0 };

		for (SElem* pCur = pHead; pCur != NULL; pCur = pCur->mpNext) {
			sprintf_s(szBuffer,	256,	"%s;%s;%d;%d;%.2lf;\n",	pCur-
>mInfo.mAuthor,	pCur->mInfo.mName,	pCur->mInfo.mPages,	pCur->mInfo.mYear,	pCur-
>mInfo.mPrice);
			fwrite(szBuffer, sizeof(char), strlen(szBuffer), fp);
		}

		printf("\n//List is written to \"%s\"//\n\n", szFileName);
		fclose(fp);
	}

}

//---------------------------------------------------------------------------

//This function outputs given list to console in the form of table.
void OutputList(SElem* pHead) {
	if (isListEmpty(pHead)) {
		printf("The list is empty. Can't print an empty list :(");
		return;
	}

	PrintStars(120);
	printf("\n*\tAuthor\t\t*\t		Book  Name\t\t\t*		Volume\t*	Published
in\t*\tPrice\t\t*\n");
	PrintStars(120);

	for (SElem* pCur = pHead;pCur!=NULL;pCur=pCur->mpNext) {
		printf("\n*%-20s\t*%-35s\t*\t%-4d\t*\t%-2d\t*\t%-.2f\t\t*\n",		pCur-
>mInfo.mAuthor,	pCur->mInfo.mName,	pCur->mInfo.mPages,	pCur->mInfo.mYear,	pCur-
>mInfo.mPrice);
	}
	PrintStars(120);
}

//---------------------------------------------------------------------------

//This function initializes list head with NULL.
void InitList(SElem** ppHead) {
	*ppHead = NULL;
```

```
}

//-----------------------------------------------------------------------------

//This function checks if the list is empty.
bool isListEmpty(SElem* pHead) {
	return NULL == pHead;
}

//-----------------------------------------------------------------------------

//This functions calculates the number of the elements in the list.
int ListElemCount(SElem* pHead) {

	int cCount = 0;
	SElem* pCur = pHead;

	while (pCur) {
		pCur = pCur->mpNext;
		++cCount;
	}

	return cCount;
}

//-----------------------------------------------------------------------------

//This function adds given element to the beginning of the list.
void AddElToBeg(SElem** ppHead, SElem* pEl) {

	if (isListEmpty(*ppHead)) {
		*ppHead = pEl;
		pEl->mpNext = NULL;
		return;
	}

	pEl->mpNext = *ppHead;
	*ppHead = pEl;
}

//-----------------------------------------------------------------------------

/*This function adds given element to the list to the position specified by user.
Includes functions AddElToBeg and AddElToEnd in it to add elements to the beginning and
the end respectively.*/
void AddElToPos(SElem** ppHead, SElem* pEl, int iPos) {

	if (iPos < 0 || iPos > ListElemCount(*ppHead)) {
		exit(2);
	}

	if (iPos == 0) {
		AddElToBeg(ppHead, pEl);
		return;
	}

	if (iPos == ListElemCount(*ppHead)) {
		AddElToEnd(ppHead, pEl);
		return;
	}

	SElem* pCur = *ppHead;

	for (int i = 1; i < iPos && pCur->mpNext; pCur = pCur->mpNext, ++i) {}
	pEl->mpNext = pCur->mpNext;
	pCur->mpNext = pEl;
}
```

```
//-----------------------------------------------------------------------------

//This function adds given element to the end of the list.
void AddElToEnd(SElem** ppHead, SElem* pEl) {

	if (isListEmpty(*ppHead)) {
		*ppHead = pEl;
		pEl->mpNext = NULL;
		return;
	}

	SElem* pCur = *ppHead;

	while (pCur->mpNext != NULL) {
		pCur = pCur->mpNext;
	}
	pCur->mpNext = pEl;
	pEl->mpNext = NULL;
}

//-----------------------------------------------------------------------------

//This function adds creates book (as a new element) with data input by user and adds
it to the end of the list.
void AddBookToEnd(SElem** ppHead, char szAuthor[50], char szName[50], int iPages, int
iYear, double dbPrice) {
	SElem* pNewEl = (SElem*)malloc(sizeof(SElem));

	if (pNewEl) {
		strcpy(pNewEl->mInfo.mAuthor, szAuthor);
		strcpy(pNewEl->mInfo.mName, szName);
		pNewEl->mInfo.mPages = iPages;
		pNewEl->mInfo.mYear = iYear;
		pNewEl->mInfo.mPrice = dbPrice;

		AddElToEnd(ppHead, pNewEl);
	}
	else exit(2);
}

//-----------------------------------------------------------------------------

//This function is made to automatize the process of adding books to the end of the
list.
void AddingBooks(SElem** ppHead) {

	printf("\n\n\t\t\t\t\t\tAdding mode.\n");

	int iFlag;

	do {

		char szAuthor[50] = { 0 };
		char szName[50] = { 0 };
		int iPages;
		int iYear;
		double dbPrice;

		SElem* pNewEl = (SElem*)malloc(sizeof(SElem));

			if (pNewEl) {
				printf("Enter the author: \n");
				scanf("%s", &szAuthor);
				strcpy(pNewEl->mInfo.mAuthor, szAuthor);

				printf("Enter the book name: \n");
				scanf("%s", &szName);
				strcpy(pNewEl->mInfo.mName, szName);
```

```
                        printf("Enter the number of pages: \n");
                        scanf("%d", &iPages);
                        pNewEl->mInfo.mPages = iPages;

                        printf("Enter the year: \n");
                        scanf("%d", &iYear);
                        pNewEl->mInfo.mYear = iYear;

                        printf("Enter the price: \n");
                        scanf("%lf", &dbPrice);
                        if (dbPrice < 0) {
                              printf("\nPrice can not be less than 0. Try another
price!\n");
                              scanf("%lf", &dbPrice);
                        }
                        pNewEl->mInfo.mPrice = dbPrice;
                  }
                  else exit(5);

            int iPos;
            printf("Enter your pos (0 - first SElement): \n");
            scanf("%d", &iPos);

            AddElToPos(ppHead, pNewEl, iPos);

            printf("Enter zero-number if you want to stop adding books to the list.
\n");
            scanf("%d", &iFlag);

      } while (iFlag);
}

//-----------------------------------------------------------------------------

//This function deletes an element in the beginning of the list.
void DelElBeg(SElem** ppHead) {

      SElem* pTemp = *ppHead;
      *ppHead = (*ppHead)->mpNext;
      free(pTemp);
}

//-----------------------------------------------------------------------------

//This function deletes an element inside the list.
void DelElInside(SElem* pHead, SElem * pDelete) {

      SElem* pCur = pHead;
      while (pCur->mpNext != pDelete) {
            pCur = pCur->mpNext;
      }
      pCur->mpNext = pCur->mpNext->mpNext;

}

//-----------------------------------------------------------------------------

//This function deletes an element in the end of the list.
void DelElEnd(SElem* pHead) {

      SElem* pCur = pHead;
      while (pCur->mpNext->mpNext != NULL) {
            pCur = pCur->mpNext;
      }
      SElem* pLast = pCur->mpNext;
      pCur->mpNext = NULL;
      free(pLast);
```

```
}

//-------------------------------------------------------------------------------

//This function deletes an element from the list at the position specified by user.
void DelElByPos(SElem** ppHead, int iPos) {
	SElem* pCur = *ppHead;

	if (iPos < 0 || iPos > ListElemCount(*ppHead)) {
		exit(2);
	}

	if (iPos == 0) {
		DelElBeg(ppHead);
		return;
	}

	if (iPos == ListElemCount(*ppHead)) {
		DelElEnd(*ppHead);
		return;
	}

	else {
		for (int i = 1; i < iPos && pCur->mpNext; pCur = pCur->mpNext, ++i) {}
		DelElInside(*ppHead, pCur);
		return;
	}
}

//-------------------------------------------------------------------------------

//This function deletes books that have name starting with letter 'P', 'K', 'L' of any
case.
void DelPKLBookName(SElem** ppHead, int cCount) {
	if (isListEmpty(*ppHead)) {
		printf("\nThe list is empty. Try reading it by typing \"1\" !");
		return;
	}

	SElem* pTemp = *ppHead;
	int iPos = 0;
	while (pTemp->mpNext!=NULL)
	{
		if (pTemp->mInfo.mName[0] == 'P' || pTemp->mInfo.mName[0] == 'K' || pTemp-
>mInfo.mName[0] == 'L' || pTemp->mInfo.mName[0] == 'p' || pTemp->mInfo.mName[0] == 'k'
|| pTemp->mInfo.mName[0] == 'l') {
			if (iPos == 0) {
				DelElBeg(ppHead);
				pTemp = *ppHead;
			}
			else if (iPos == cCount) {
				DelElEnd(*ppHead);
			}
			else {
				DelElInside(*ppHead, pTemp);
			}
		}
		pTemp = pTemp->mpNext;
		++iPos;
	}
}

//-------------------------------------------------------------------------------

//This function is made to automatize the process of deleting books from the list at
the positions input by user.
void DeletingBooks(SElem** ppHead) {
	if (isListEmpty(*ppHead)) {
```

```
		printf("\nThe list is empty. Try reading it by typing \"1\" !");
		return;
	}

	printf("\n\n\t\t\t\t\t\t\tDeleting mode.\n");

	int iFlag;

	do {

		int iPos;
		printf("Enter your pos (0 - first SElement): \n");
		scanf("%d", &iPos);

		DelElByPos(ppHead, iPos);

		printf("Enter zero-number if you want to stop deleting books from the list.
\n");
		scanf("%d", &iFlag);

	} while (iFlag);

}

//-------------------------------------------------------------------------------

//This function swaps the first and the second elements of the list.
void SwapElemsBeg(SElem** ppHead) {

	SElem* pEl = (*ppHead)->mpNext;
	(*ppHead)->mpNext = pEl->mpNext;
	pEl->mpNext = *ppHead;
	*ppHead = pEl;
}

//-------------------------------------------------------------------------------
//This function swaps two sequential elements inside the list.
void SwapElemsInside(SElem* pHead, SElem* pEl) {

	SElem* pCur = pHead;
	for (; pCur->mpNext != pEl; pCur = pCur->mpNext) {}
	SElem* pEl1 = pCur->mpNext;
	SElem* pEl2 = pCur->mpNext->mpNext;

	pEl1->mpNext = pEl2->mpNext;
	pEl2->mpNext = pEl1;
	pCur->mpNext = pEl2;
 }

//-------------------------------------------------------------------------------

//This function swaps the last and the second-to-last elements of the list.
void SwapElemsEnd(SElem* pHead) {

	SElem* pCur = pHead;
	for (; pCur->mpNext->mpNext->mpNext != NULL; pCur = pCur->mpNext) {}

	SElem* pEl = pCur->mpNext;
	SElem* pLast = pCur->mpNext->mpNext;

	pCur->mpNext = pLast;
	pLast->mpNext = pEl;
	pEl->mpNext = NULL;

}

//-------------------------------------------------------------------------------
```

```
//This function sorts the list of the books by their name in alphabetic order.
void ListSortByName(SElem** ppHead) {
	if (isListEmpty(*ppHead)) {
		printf("\nThe list is empty. Try reading it by typing \"1\" !");
		return;
	}

	for (int i = 0; i < ListElemCount(*ppHead); ++i) {
		int iCount = 1;
		SElem* pCur = *ppHead;

		while (iCount < ListElemCount(*ppHead)-1) {
			SElem* pEl1 = pCur;
			SElem* pEl2 = pCur->mpNext;

			if (strcmp(pEl1->mInfo.mName, pEl2->mInfo.mName) > 0) {

				if (iCount == 1) {
					SwapElemsBeg(ppHead);
				}

				else if (iCount == ListElemCount(*ppHead)) {
					SwapElemsEnd(*ppHead);
				}
				else {
					SwapElemsInside(*ppHead, pEl1);
					pCur = pEl2;
				}
			}
			pCur = pCur->mpNext;
			++iCount;
		}
	}
}

//-------------------------------------------------------------------------------  !

//This function creates sublist of books which are above average of the list given in
price.
int AboveAveragePriceBooklist(SElem* pHead, SElem** ppHead2) {
	bool bListCreated = false;

	if (isListEmpty(pHead)) {
		printf("\nThe list is empty. Try reading it by typing \"1\" !");
		return;
	}

	SElem* pCur = pHead;
	double dbAvg = 0;

	for (; pCur != NULL; pCur = pCur->mpNext) {
		dbAvg += pCur->mInfo.mPrice;
	}

	dbAvg /= ListElemCount(pHead);

	pCur = pHead;

	do {
		if (pCur && pCur->mInfo.mPrice >= dbAvg) {
			AddBookToEnd(ppHead2, pCur->mInfo.mAuthor, pCur->mInfo.mName, pCur-
>mInfo.mPages, pCur->mInfo.mYear, pCur->mInfo.mPrice);
			bListCreated = true;
		}
		pCur = pCur->mpNext;
	} while (pCur && pCur->mpNext != NULL);
	return bListCreated;
}
```

```
//---------------------------------------------------------------------------

//This function frees dynamically allocated memory of the list.
void FreeList(SElem **ppHead) {
	if (isListEmpty(*ppHead)) {
		return;
	}

	SElem* pCur = *ppHead;

	while (pCur) {
		*ppHead = pCur->mpNext;
		free(pCur);
		pCur = *ppHead;
	}

	*ppHead = NULL;
}

//---------------------------------------------------------------------------

//This function prints '*' symbols in a given quantity.
void PrintStars(int iNum) {
	for (int i = 0; i < iNum; ++i) {
		putchar('*');
	}
}

//---------------------------------------------------------------------------

//This function prints 'x' symbols in a given quantity.
void PrintX(int iNum) {
	for (int i = 0; i < iNum; ++i) {
		putchar('x');
	}
}

//---------------------------------------------------------------------------

//This function prints the graphical representation of the program menu.
void PrintMenu() {
	PrintX(120);
	printf("x\t\t\t\t\t\t\t//BooksDB::List//\t\t\t\t\t\tx\n");
	PrintX(120);
	printf("x\tType number needed to do the action and enjoy the show
:)\t\t\t\t\t\tx\n");
	PrintX(120);
	printf("x\t1\tx\tRead list from file\t\t\t\t\t\t\t\tx\n");
	printf("x\t2\tx\tWrite list to file\t\t\t\t\t\t\tx\n");
	printf("x\t3\tx\tAdding books\t\t\t\t\t\t\tx\n");
	printf("x\t4\tx\tDeleting books\t\t\t\t\t\t\tx\n");
	printf("x\t5\tx\tSort list\t\t\t\t\t\t\tx\n");
	printf("x\t6\tx\tSpecial: Delete PKL\t\t\t\t\t\tx\n");
	printf("x\t7\tx\tSpecial: Sort above average\t\t\t\t\tx\n");
	PrintX(120);
	printf("\nx\t8\tx\tExit\t\t\t\t\t\t\t\tx\n");
	PrintX(120);
}

//---------------------------------------------------------------------------

//This function implements the menu of the program. A user can choose an action they
need and it will call a corresponding function.
void Menu(SElem** ppHead) {
	PrintMenu();

	int iOption = 1;
```

```
	while (iOption != 8) {
		printf("\n\n//  ");
		scanf("%d", &iOption);

		if (iOption == 1) {
			ReadListFromFile(ppHead, "input.txt");
			OutputList(*ppHead);
			printf("\n\n...function ended its work\n\n");
			PrintMenu();
		}

		else if (iOption == 2) {
			WriteListToFile(*ppHead, "output");
			printf("\n\n...function ended its work\n\n");
			PrintMenu();
		}

		else if (iOption == 3) {
			AddingBooks(ppHead);
			printf("List after adding new books: \n");
			OutputList(*ppHead);
			printf("\n\n...function ended its work\n\n");
			PrintMenu();
		}

		else if (iOption == 4) {
			DeletingBooks(ppHead);
			printf("List after deleting chosen books: \n");
			OutputList(*ppHead);
			printf("\n\n...function ended its work\n\n");
			PrintMenu();
		}

		else if (iOption == 5) {
			if (isListEmpty(*ppHead)) {
				printf("\nThe list is empty. Try reading it by typing \"1\"
!\n\n");
				PrintMenu();
				continue;
			}

			ListSortByName(ppHead);
			printf("\n\nSorted list: \n\n");
			OutputList(*ppHead);
			printf("\n\n...function ended its work\n\n");
			PrintMenu();
		}

		else if (iOption == 6) {
			if (isListEmpty(*ppHead)) {
				printf("\nThe list is empty. Try reading it by typing \"1\"
!\n\n");
				PrintMenu();
				continue;
			}

			DelPKLBookName(ppHead, ListElemCount(*ppHead));
			printf("List after deleting books if their name is starting with
P,K,L: \n");
			OutputList(*ppHead);
			printf("\n\n...function ended its work\n\n");
			PrintMenu();
		}

		else if (iOption == 7) {
			if (isListEmpty(*ppHead)) {
```

```
                        printf("\nThe list is empty. Try reading it by typing \"1\"
!\n\n");
                        PrintMenu();
                        continue;
                }

                SElem* pHead2;
                InitList(&pHead2);
                if (AboveAveragePriceBooklist(*ppHead, &pHead2)) {
                        printf("\n\nAbove average price book list: \n\n");
                        OutputList(pHead2);

                        ListSortByName(&pHead2);
                        printf("\n\nSorted list: \n\n");
                        OutputList(pHead2);
                        printf("\n\n...function ended its work\n\n");
                        PrintMenu();
                }
                else {
                        printf("\n\nError creating new list...\n");
                }
        }

        else if (iOption == 8) {
                FreeList(ppHead);
                printf("\n\n\t\t\t\t\t\tThe list is cleansed!");
                printf("\n\n\t\t\t\t\t\tExiting program... \n\n");
                exit(0);
        }

        else {
                printf("\nOption #%d does not yet exist! Try another one.\n",
iOption);
        }

    }
}

//---------------------------------------------------------------------------
```

main.c

```
#include "func.h"

int main() {

    SElem* pHead=NULL;
    InitList(&pHead);

    Menu(&pHead);

    return 0;
}
```

ВИСНОВКИ

Виконуючи цю лабораторну роботу, я познайомився із важливими етапами ЖЦ ПЗ – власне етапами проектування та реалізації. Також попрацював над алгоритмами та структурами даних – поведінковою та структурною складовими результатів проектування ПЗ відповідно. Оформив усю необхідну документацію до моєї програми (книжкової бази даних) відповідно до завдань лабораторної

роботи: оглянув свою програму з точки зору архітектури – використаних структур даних, покрокових алгоритмів. Перед тим привів свою програму до модульної структури та відформатував її текст відповідно до правил оформлення коду на C++.