Міністерство освіти і науки, молоді та спорту України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 2**

«Документування етапів проектування та кодування програми»

**з дисципліни «Вступ до інженерії програмоного забезпечення»**

**Лектор:**

доцент кафедри ПЗ

Левус Є.В.

**Виконав:**

студ. групи ПЗ-16

Чаус І.В.

**Прийняв:**

асистент кафедри ПЗ

Самбір А.А.

«\_\_\_» \_\_\_\_\_\_ 2020 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2020

**Тема:** документація етапів проектування та кодування програми.

**Мета:** навчитися документувати основні результати етапів проектування та кодування найпростіших програм.

**Хід роботи**

**28. Як записуються класи та їх складові у мові С++?**

Назва класу завжди починаємо писати з великої літери С. Обов’язково повинен бути коментар який описує властивості класу.

В класах рекомендовано використовувати таку послідовність методів у кожній секції (private, protected, public) класу:

* Статичні
* Конструктори
* Деструктори
* Змінні-члени
* Оператори
* Методи члени

**16. Скільки входів та виходів має блок циклу? Відповідь пояснити.**

Блок циклу має два входи і два виходи. Один вхід використовується, коли програма переходить до виконання циклу, другий вхід зображає повторну перевірку чи виконується умова циклу, перший вихід зображає вихід з циклу і продовження виконання програми, другий вихід зображає виконання умови циклу і перехід до виконання тіла циклу.

**10. Що таке алгоритм? Навести приклад.**

Алгоритм — це набір інструкцій, які описують порядок дій виконавця, щоб досягти результату розв’язання задачі за скінченну кількість дій. Прикладом може слугувати розв’язування квадратного рівняння в математиці, де для того, щоб знайти корені рівняння треба спочатку знайти дискримінант, якщо дискримінант більший рівний нулю, підставити його у рівняння для знаходження коренів і таким чином знайти корені даного квадратного рівняння.

**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 ;

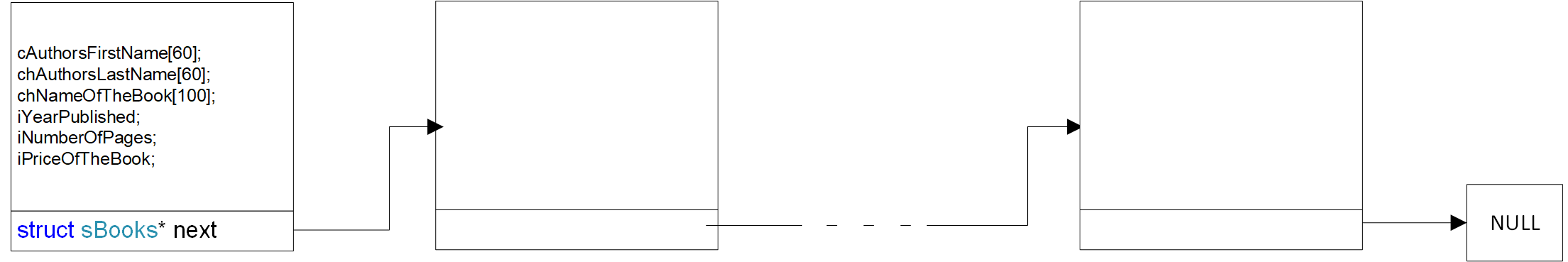
2. блок-схема алгоритмів – основної функції й двох окремих функційпідпрограм (наприклад, сортування та редагування);

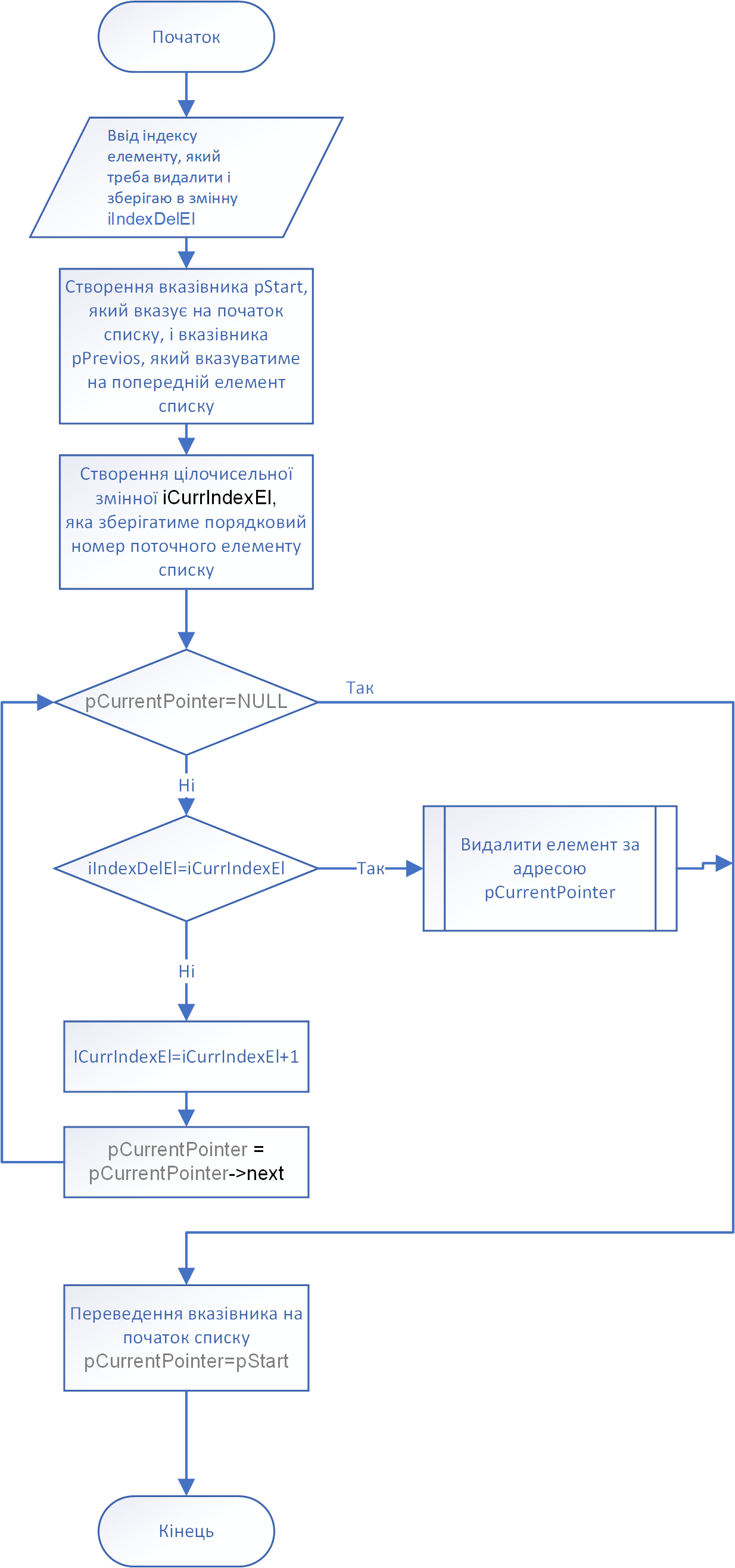
3. текст програми з коментарями та оформлений згідно вище наведених рекомендацій щодо забезпечення читабельності й зрозумілості. Для схематичного зображення структур даних, блок-схеми алгоритму використати редактор MS-Visio.

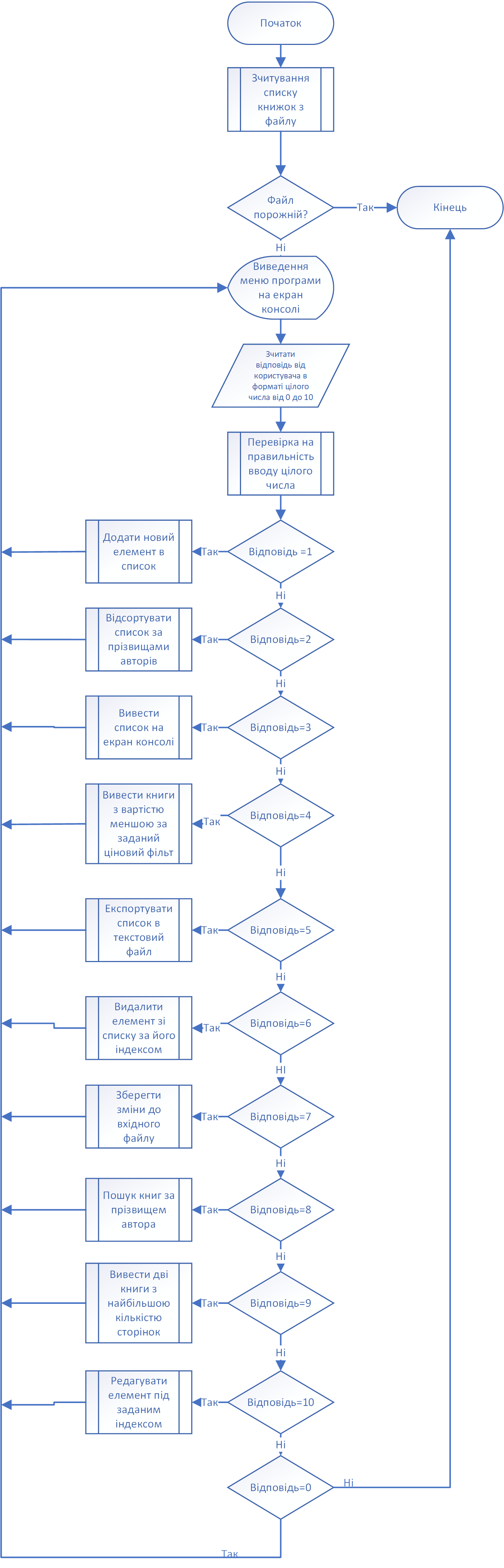
**Частина ІІІ.** У редакторі MS-Visio розробити зразки фігур, які були використані для схематичного зображення структур даних програм, як готові трафарети до використання . Сформувати свою бібліотеку фігур – окремий користувацький файл із використаними зразками.

**Отримані результати**

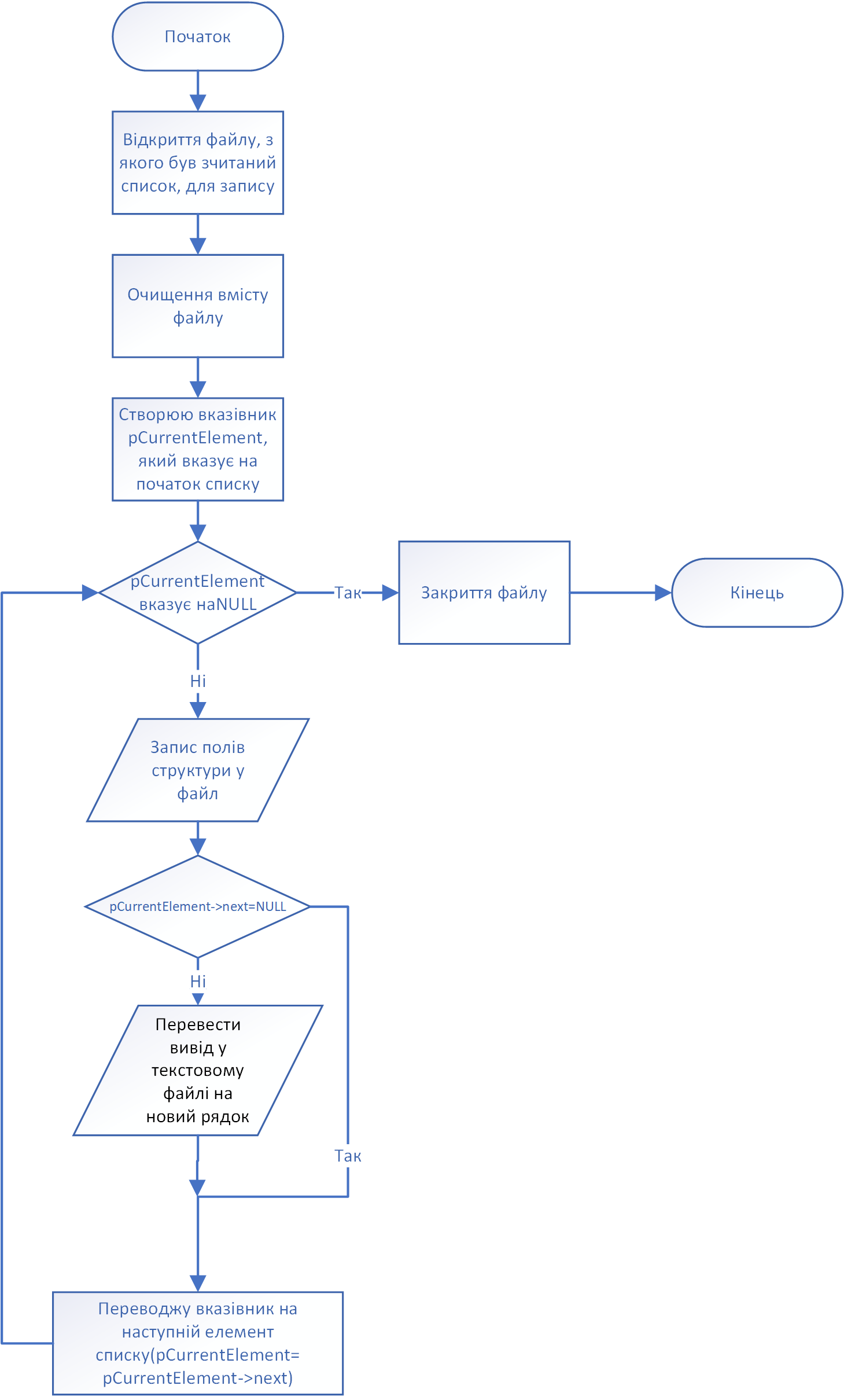
1. Схематична зображення структур даних, які використовуються для збереження інформації.

2. Блок-схеми алгоритмів

Функція-підпрограма видалення елементу за індексом

Основна функція

Функція-підпрограма збереження змін у вхідному файлі



**3. Текст програми**

**books.h**

**//структура sBooks назначена для збереження списку книг в білбліотеці/книгарні**

//у вигляді однозв'язного списку. Містить поля для збереження імені автора

//(до 59 символів, без пробілів), прізвища автора(до 59 символів, без пробілів)

//назви книги(до 99 символів), року видання книги(цілочисельне додатнє значення

//формату рррр), кількості сторінок в книзі(цілочисельне додатнє значення),

//вартість книги(цілочисельне невід'ємне значення) та вказівник на наступній

//елемент

struct sBooks {

char chAuthorsFirsName[60];

char chAuthorsLastName[60];

char chNameOfTheBook[100];

int iYearPublished;

int iNumberOfPages;

double iPriceOfTheBook;

struct sBooks\* pNext;

};

//виводить смужку для створення таблиці

//приймає єдиниц цілочисельний параметр, який вказує на кількість символів "-"

void vPrintLine(int);

//сортує список за прізвищами авторів в алфавітному порядку

//приймає єдиний параметр - вказівник на структуру Sbooks, який вказує на

//початок списку

void vSortList(struct sBooks\*);

//перевіряє чи відповідає стрічка формату імені/прізвища автора(лише літери, без

//пробілів)

//приймає один параметр - вказівник на стрічку-буфер, яку треба перевірити

//якщо стрічка задовільняє формат повертає 1, в іншому випадку - 0

int nIsString(char\*);

//перевіряє чи є дійсне число корректно введеним(без літер, додатнє)

//приймає один параметр - вказівник на стрічку-буфер, яку треба перевірити

//якщо стрічка є відображенням дійсного числа повертає 1, в іншому випадку - 0

int nIsRealNumber(char\*);

//перевіряє чи є ціле число корректно введеним(без літер, додатнє)

//приймає один параметр - вказівник на стрічку-буфер, яку треба перевірити

//якщо стрічка є відображенням дійсного числа повертає 1, в іншому випадку - 0

int nIsNum(char\*);

//додає новий елемент в кінець списку

//приймає один параметр - вказівник на вказівник(Sbooks) початку списку

void vAddNewNode(struct sBooks\*\*);

//зчитує список книг з файлу і динамічно виділяє пам'ять для елементів списку

//параметрами приймає 1)вказівник на стрічку(адреса файлу який треба зчитати)

// 2)вказівник на вказівник першого елементу списку

//повертає 0, якщо файл є пустим

// 1, якщо зчитування і виділення пам'яті були успішними

int nReadFromFile(char\*, struct sBooks\*\*);

//видаляє елемент за заданою адресою і звільняє пам'ять динамічно виділену для

//нього

//параметрами приймає: 1)вказівник на вказівник на адресу елементу, який треба

//видалити

// 2)вказівник на вказівник на адресу елементу, який знаходиться перед

// елементом, який треба

//видалити 3)вказівник на вказівник початку списку

void vRemoveElement(struct sBooks\*\*, struct sBooks\*\*, struct sBooks\*\*);

//виводить книги з вартістю меншою за задану з можливістю їх видалення зі списку

//у випадку, якщо всі книги дорожчі за задану вартість, на екран виводиться

//повідомлення

//приймає єдиний параметр- вказівник на вказівник початку списку

void vLessThanCertainPrice(struct sBooks\*\*);

//виводить список у вигляді таблиці на екран консолі

//параметром приймає вказівник на перший елемент списку

void vPrintList(struct sBooks\*);

//зберігає список в файл, адреса файлу вводиться користувачем під час виконання

//функції

//парамтером приймає вказівник на початок списку

void vWriteToFile(struct sBooks\*);

//видаляє елемент за його індексом

//якщо елемента за взазаним індексом не існує, виводить на екран повідомлення

//про це

//параметром приймає вказівник на вказівник початку списку

void vDeleteSingleElement(struct sBooks\*\*);

//зберігає зміни в списку до файлу з якого список був зчитаний

//параметром приймає вказівник на початок списку

void vSaveChanges(struct sBooks\*);

//проводить пошук за прізвищем автора, яке задається користувачем в ході

//виконання функції

//якщо не знайдено елементів, які задовільняють умови, на екран виводиться

//відповідне повідомлення

//параметром приймає вказівник на вказівник початку списку

void vSearchByAuthorsLastName(struct sBooks\*\*);

//перевіряє чи є файл порожнім

//параметром приймає вказівник на файловий потік

//якщо файл є порожнім, повертає 1. В іншому випадку - 0

int nIfFileEmpty(FILE\*);

//знаходить дві книги з найбільшою кількістю сторінок і виводить їх на екран

//консолі

//параметром приймає вказівник на початку списку

void vBooksWithTheMostPages(struct sBooks\*);

//надає можливість користувачеві редагувати елемент за заданою адресою

//параметром приймає вказівник на елемент, який потрібно відредагувати

void vEditElement(struct sBooks\*);

//надає можливість користувачеві редагувати елемент за заданим індексом, який

//задається під час виконання функції

//параметром приймає вказівник на початок списку

void vEditElementByIndex(struct sBooks\*);

//зчитування дійсного додатнього значення з клавіатури, присутній захисть від

//некоректного вводу

//повертає введене користувачем значення

double dReadFoolProofDouble();

//зчитування цілого додатнього значення з клавіатури, присутній захисть від

//некоректного вводу

//повертає введене користувачем значення

int iReadFoolProofInt();

//зчитування стрічки, яка відповідає формату ім'я/прізвище

//параметром приймає вказівник на зчитувану стрічку

**void vReadFoolProofNameString(char\* testedString);**

**books.c**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <ctype.h>**

**#include <string.h>**

**#include <windows.h>**

**#include "books.h"**

**#define NUMBER\_OF\_LINES \**

**115 //значення ширини роздільної лініїї в таблицях**

**//виводить елемент списку у вигляді рядку таблиці на екран консолі**

**//приймає два параметри: вказівник на елемент, який треба видрукувати**

**//та порядковий номер**

**void printComponent(struct sBooks\* pSCurrentBooksPointer, int iListOrder) {**

**printf("| %2d|%14s|%15s|%49s| %6d| %8d| %7.2lf |\n", iListOrder,**

**pSCurrentBooksPointer->chAuthorsFirsName,**

**pSCurrentBooksPointer->chAuthorsLastName,**

**pSCurrentBooksPointer->chNameOfTheBook,**

**pSCurrentBooksPointer->iYearPublished,**

**pSCurrentBooksPointer->iNumberOfPages,**

**pSCurrentBooksPointer->iPriceOfTheBook);**

**vPrintLine(NUMBER\_OF\_LINES);**

**}**

**//------------------------------------------------------------------------------**

**//виводить заголовок таблиці на консоль**

**void printHeader() {**

**vPrintLine(NUMBER\_OF\_LINES);**

**printf(**

**"| # | FIRST NAME | LAST NAME | "**

**"NAME OF THE BOOK | YEAR | PAGES | PRICE |\n");**

**vPrintLine(NUMBER\_OF\_LINES);**

**}**

**//------------------------------------------------------------------------------**

**int nIsString(char chTestedString[]) {**

**if (strlen(chTestedString) > 1) {**

**for (int i = 0; i < strlen(chTestedString); i++) {**

**if (isalpha(chTestedString[i]) == 0) {**

**return 0;**

**}**

**}**

**return 1;**

**}**

**return 0;**

**}**

**//------------------------------------------------------------------------------**

**int nIsRealNumber(char chTestedNumberString[]) {**

**if (strlen(chTestedNumberString) > 0) {**

**for (int i = 0; i < strlen(chTestedNumberString); i++) {**

**if (isdigit(chTestedNumberString[i]) == 0 &&**

**chTestedNumberString[i] != '.') {**

**return 0;**

**}**

**}**

**return 1;**

**}**

**return 0;**

**}**

**//------------------------------------------------------------------------------**

**int nIsNum(char chTestedNumberString[]) {**

**if (strlen(chTestedNumberString) > 0) {**

**for (int i = 0; i < strlen(chTestedNumberString); i++) {**

**if (isdigit(chTestedNumberString[i]) == 0) {**

**return 0;**

**}**

**}**

**return 1;**

**}**

**return 0;**

**}**

**//------------------------------------------------------------------------------**

**void vAddNewNode(struct sBooks\*\* pSCurrentBooksPointer) {**

**struct sBooks\* pSStartingPointer = \*pSCurrentBooksPointer;**

**char сBuffer[50];**

**while ((\*pSCurrentBooksPointer)->pNext) {**

**\*pSCurrentBooksPointer = (\*pSCurrentBooksPointer)->pNext;**

**}**

**(\*pSCurrentBooksPointer)->pNext = (sBooks\*)malloc(sizeof(sBooks));**

**(\*pSCurrentBooksPointer) = (\*pSCurrentBooksPointer)->pNext;**

**(\*pSCurrentBooksPointer)->pNext = NULL;**

**printf("Please enter new entry\n");**

**printf("Enter author's first name:");**

**vReadFoolProofNameString((\*pSCurrentBooksPointer)->chAuthorsFirsName);**

**getchar();**

**printf("Enter author's last name: ");**

**vReadFoolProofNameString((\*pSCurrentBooksPointer)->chAuthorsLastName);**

**printf("Enter name of the book: ");**

**getchar();**

**scanf("%[^\n]s", &(\*pSCurrentBooksPointer)->chNameOfTheBook);**

**while (strlen((\*pSCurrentBooksPointer)->chNameOfTheBook) < 2) {**

**printf("You cannot skip this step. Please enter the name of the book:\t");**

**getchar();**

**scanf("%[^\n]s", &(\*pSCurrentBooksPointer)->chNameOfTheBook);**

**}**

**printf("Enter the year: ");**

**getchar();**

**(\*pSCurrentBooksPointer)->iYearPublished = iReadFoolProofInt();**

**printf("Enter the number of pages: ");**

**(\*pSCurrentBooksPointer)->iNumberOfPages = iReadFoolProofInt();**

**printf("Enter the price: ");**

**(\*pSCurrentBooksPointer)->iPriceOfTheBook = dReadFoolProofDouble();**

**\*pSCurrentBooksPointer = pSStartingPointer;**

**printf("Do you want to resort the list: yes(1) no(0) ");**

**if (iReadFoolProofInt() == 1) {**

**vSortList(pSStartingPointer);**

**printf("List is resorted\n");**

**}**

**}**

**//------------------------------------------------------------------------------**

**int nReadFromFile(char\* cFileOutPutAddress, struct sBooks\*\* pCurrentElement) {**

**FILE\* pInputFile;**

**pInputFile = fopen(cFileOutPutAddress, "r");**

**if (nIfFileEmpty(pInputFile) == 1) {**

**printf(**

**"The file is emptry. Please add some information"**

**"to the list first(at least 1 item)\n");**

**return 0;**

**}**

**struct sBooks\* pPreviousElement = NULL;**

**struct sBooks\* pStartingElement;**

**\*pCurrentElement = (struct sBooks\*)malloc(sizeof(struct sBooks));**

**pStartingElement = \*pCurrentElement;**

**while (getc(pInputFile) != EOF) {**

**pPreviousElement = \*pCurrentElement;**

**if (!pStartingElement) {**

**pStartingElement = \*pCurrentElement;**

**}**

**fscanf(pInputFile, "%s %s %c %[^\"]s",**

**&(\*pCurrentElement)->chAuthorsFirsName,**

**&(\*pCurrentElement)->chAuthorsLastName,**

**&(\*pCurrentElement)->chNameOfTheBook,**

**&(\*pCurrentElement)->chNameOfTheBook);**

**fscanf(pInputFile, "%\*c%d", &(\*pCurrentElement)->iYearPublished);**

**fscanf(pInputFile, "%\*c%d", &(\*pCurrentElement)->iNumberOfPages);**

**fscanf(pInputFile, "%\*c%lf", &(\*pCurrentElement)->iPriceOfTheBook);**

**(\*pCurrentElement)->pNext = (struct sBooks\*)malloc(sizeof(struct sBooks));**

**(\*pCurrentElement) = (\*pCurrentElement)->pNext;**

**}**

**free(\*pCurrentElement);**

**pPreviousElement->pNext = NULL;**

**\*pCurrentElement = pStartingElement;**

**fclose(pInputFile);**

**return 1;**

**}**

**//------------------------------------------------------------------------------**

**void vRemoveElement(struct sBooks\*\* pCurrentBook, struct sBooks\*\* pPreviousBook,**

**struct sBooks\*\* pStartBook) {**

**struct sBooks\* pTempBook = NULL;**

**if (!(\*pPreviousBook)) {**

**pTempBook = \*pCurrentBook;**

**\*pCurrentBook = (\*pCurrentBook)->pNext;**

**\*pStartBook = \*pCurrentBook;**

**free(pTempBook);**

**\*pPreviousBook = NULL;**

**} else {**

**(\*pPreviousBook)->pNext = (\*pCurrentBook)->pNext;**

**free(\*pCurrentBook);**

**\*pCurrentBook = \*pPreviousBook;**

**}**

**}**

**//------------------------------------------------------------------------------**

**void vLessThanCertainPrice(struct sBooks\*\* pListStart) {**

**printf("Please enter the price filter:");**

**double dPriceFilter = dReadFoolProofDouble();**

**struct sBooks\* pCurrentBook = \*pListStart;**

**printHeader();**

**struct sBooks\* start = pCurrentBook;**

**int iElementIndex = 1;**

**while (pCurrentBook) {**

**if (pCurrentBook->iPriceOfTheBook < dPriceFilter) {**

**printComponent(pCurrentBook, iElementIndex);**

**iElementIndex++;**

**}**

**pCurrentBook = pCurrentBook->pNext;**

**}**

**if (iElementIndex > 1) {**

**printf(**

**"Do you want to remove these"**

**" elements from the list?: yes(1), no(0):");**

**int answer;**

**answer = iReadFoolProofInt();**

**if (answer == 1) {**

**struct sBooks\* pTempListStart = \*pListStart;**

**struct sBooks\* pPreviousBook = NULL;**

**while (\*pListStart) {**

**if ((\*pListStart)->iPriceOfTheBook < dPriceFilter) {**

**vRemoveElement(pListStart, &pPreviousBook, &pTempListStart);**

**} else {**

**pPreviousBook = \*pListStart;**

**\*pListStart = (\*pListStart)->pNext;**

**}**

**}**

**\*pListStart = pTempListStart;**

**printf(**

**"All items with a price of less than %.2lf UAH "**

**"were removed successfully \n",**

**dPriceFilter);**

**}**

**} else {**

**printf(**

**"| All books cost more than %.2lf UAH"**

**" | \n",**

**dPriceFilter);**

**vPrintLine(NUMBER\_OF\_LINES);**

**printf("Press enter to continute");**

**}**

**}**

**//------------------------------------------------------------------------------**

**void vPrintList(struct sBooks\* pListStart) {**

**system("cls");**

**printHeader();**

**int iElementIndex = 1;**

**while (pListStart) {**

**printComponent(pListStart, iElementIndex);**

**pListStart = pListStart->pNext;**

**iElementIndex++;**

**}**

**}**

**//------------------------------------------------------------------------------**

**void vPrintLine(int iNumberOfLines) {**

**for (int i = 0; i < iNumberOfLines; i++) printf("-");**

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

**}**

**//------------------------------------------------------------------------------**

**void vSortList(struct sBooks\* pListStart) {**

**char cTempAuthorsFirstName[50];**

**char cTempAuthorsLastName[50];**

**char cTempNameOfBook[60];**

**int iTempYearPublished;**

**int iTempNumberOfPages;**

**double sprice;**

**struct sBooks\* pMovingBook = pListStart;**

**while (pListStart) {**

**pMovingBook = pListStart->pNext;**

**while (pMovingBook) {**

**if (strcmp((pListStart)->chAuthorsLastName,**

**((pMovingBook)->chAuthorsLastName)) > 0) {**

**strcpy(cTempAuthorsFirstName, pListStart->chAuthorsFirsName);**

**strcpy(cTempAuthorsLastName, pListStart->chAuthorsLastName);**

**strcpy(cTempNameOfBook, pListStart->chNameOfTheBook);**

**iTempYearPublished = pListStart->iYearPublished;**

**iTempNumberOfPages = pListStart->iNumberOfPages;**

**sprice = pListStart->iPriceOfTheBook;**

**strcpy(pListStart->chAuthorsFirsName, pMovingBook->chAuthorsFirsName);**

**strcpy(pListStart->chAuthorsLastName, pMovingBook->chAuthorsLastName);**

**strcpy(pListStart->chNameOfTheBook, pMovingBook->chNameOfTheBook);**

**pListStart->iYearPublished = pMovingBook->iYearPublished;**

**pListStart->iNumberOfPages = pMovingBook->iNumberOfPages;**

**pListStart->iPriceOfTheBook = pMovingBook->iPriceOfTheBook;**

**strcpy(pMovingBook->chAuthorsFirsName, cTempAuthorsFirstName);**

**strcpy(pMovingBook->chAuthorsLastName, cTempAuthorsLastName);**

**strcpy(pMovingBook->chNameOfTheBook, cTempNameOfBook);**

**pMovingBook->iYearPublished = iTempYearPublished;**

**pMovingBook->iNumberOfPages = iTempNumberOfPages;**

**pMovingBook->iPriceOfTheBook = sprice;**

**}**

**pMovingBook = pMovingBook->pNext;**

**}**

**pListStart = pListStart->pNext;**

**}**

**}**

**//------------------------------------------------------------------------------**

**void vWriteToFile(struct sBooks\* pListStart) {**

**char cWritingPath[50];**

**printf("Enter a path to the file you want to write your data to: ");**

**getchar();**

**scanf("%[^\n]s", cWritingPath);**

**FILE\* pOutputFile;**

**pOutputFile = fopen(cWritingPath, "w");**

**fprintf(pOutputFile, " ");**

**while (pListStart) {**

**fprintf(pOutputFile, "%10s %10s \"%40s\" %4d %4d %6.2lf",**

**pListStart->chAuthorsFirsName, pListStart->chAuthorsLastName,**

**pListStart->chNameOfTheBook, pListStart->iYearPublished,**

**pListStart->iNumberOfPages, pListStart->iPriceOfTheBook);**

**if (pListStart->pNext) {**

**fprintf(pOutputFile, "\n");**

**}**

**pListStart = pListStart->pNext;**

**}**

**fclose(pOutputFile);**

**}**

**//------------------------------------------------------------------------------**

**void vDeleteSingleElement(struct sBooks\*\* pCurrentPointer) {**

**int iIndexDelEl;**

**printf("Enter elements index that you want to delete:");**

**iIndexDelEl = iReadFoolProofInt();**

**int iCurrIndexEl = 1;**

**struct sBooks\* pListStart = \*pCurrentPointer;**

**struct sBooks\* pPreviousBook = NULL;**

**while (\*pCurrentPointer) {**

**if (iCurrIndexEl == iIndexDelEl) {**

**vRemoveElement(pCurrentPointer, &pPreviousBook, &pListStart);**

**printf("Element with index number %d has been successfully deleted.\n",**

**iIndexDelEl);**

**break;**

**}**

**pPreviousBook = \*pCurrentPointer;**

**\*pCurrentPointer = (\*pCurrentPointer)->pNext;**

**iCurrIndexEl++;**

**if (!\*pCurrentPointer) {**

**printf("There is no element with index %d.\n", iIndexDelEl);**

**}**

**}**

**\*pCurrentPointer = pListStart;**

**}**

**//------------------------------------------------------------------------------**

**void vSaveChanges(struct sBooks\* pCurrentElement) {**

**FILE\* pSaveFile;**

**pSaveFile = fopen("input.txt", "w");**

**fprintf(pSaveFile, " ");**

**while (pCurrentElement) {**

**fprintf(pSaveFile, "%10s %10s \"%40s\" %4d %4d %6.2lf",**

**pCurrentElement->chAuthorsFirsName,**

**pCurrentElement->chAuthorsLastName,**

**pCurrentElement->chNameOfTheBook, pCurrentElement->iYearPublished,**

**pCurrentElement->iNumberOfPages, pCurrentElement->iPriceOfTheBook);**

**if (pCurrentElement->pNext) {**

**fprintf(pSaveFile, "\n");**

**}**

**pCurrentElement = pCurrentElement->pNext;**

**}**

**fclose(pSaveFile);**

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

**}**

**//------------------------------------------------------------------------------**

**void vSearchByAuthorsLastName(struct sBooks\*\* pListStart) {**

**char cSearchedLastName[60] = "";**

**printf("Please enter author's last name to find books written by him/her:");**

**vReadFoolProofNameString(cSearchedLastName);**

**struct sBooks\* pCurrentBook = \*pListStart;**

**printHeader();**

**int i = 1;**

**int iBookCounter = 0;**

**while (pCurrentBook) {**

**if (strcmp(pCurrentBook->chAuthorsLastName, cSearchedLastName) == 0) {**

**printComponent(pCurrentBook, i);**

**iBookCounter++;**

**}**

**pCurrentBook = pCurrentBook->pNext;**

**i++;**

**}**

**if (iBookCounter > 1) {**

**printf(**

**"Do you want to remove these elements from the list?: yes(1), "**

**"no(0):");**

**getchar();**

**int answer = iReadFoolProofInt();**

**if (answer == 1) {**

**struct sBooks\* start = \*pListStart;**

**struct sBooks\* prev = NULL;**

**while (\*pListStart) {**

**if (strcmp((\*pListStart)->chAuthorsLastName, cSearchedLastName) == 0) {**

**vRemoveElement(pListStart, &prev, &start);**

**}**

**prev = \*pListStart;**

**\*pListStart = (\*pListStart)->pNext;**

**}**

**\*pListStart = start;**

**printf("All items written by %s were deleted\n", cSearchedLastName);**

**}**

**} else {**

**printf(**

**"| %14s wrote none of the books in the list"**

**" | \n",**

**cSearchedLastName);**

**vPrintLine(NUMBER\_OF\_LINES);**

**getchar();**

**}**

**}**

**//------------------------------------------------------------------------------**

**int nIfFileEmpty(FILE\* pCheckedFile) {**

**long savedOffset = ftell(pCheckedFile);**

**fseek(pCheckedFile, 0, SEEK\_END);**

**if (ftell(pCheckedFile) == 0) {**

**return 1;**

**}**

**fseek(pCheckedFile, savedOffset, SEEK\_SET);**

**return 0;**

**}**

**//------------------------------------------------------------------------------**

**void vBooksWithTheMostPages(struct sBooks\* pListStart) {**

**struct sBooks\* pbookWithTheMostPages = pListStart;**

**struct sBooks\* pbookWithSecondToMostPages = pListStart;**

**int iLargestBooksIndex[2] = {1, 1};**

**int iCurrentBookIndex = 0;**

**while (pListStart) {**

**iCurrentBookIndex++;**

**if (pListStart->iNumberOfPages >**

**pbookWithSecondToMostPages->iNumberOfPages) {**

**if (pListStart->iNumberOfPages > pbookWithTheMostPages->iNumberOfPages) {**

**pbookWithSecondToMostPages = pbookWithTheMostPages;**

**pbookWithTheMostPages = pListStart;**

**iLargestBooksIndex[1] = iLargestBooksIndex[0];**

**iLargestBooksIndex[0] = iCurrentBookIndex;**

**} else {**

**pbookWithSecondToMostPages = pListStart;**

**iLargestBooksIndex[1] = iCurrentBookIndex;**

**}**

**}**

**pListStart = pListStart->pNext;**

**}**

**printHeader();**

**printComponent(pbookWithTheMostPages, iLargestBooksIndex[0]);**

**printComponent(pbookWithSecondToMostPages, iLargestBooksIndex[1]);**

**}**

**//------------------------------------------------------------------------------**

**void vEditElement(struct sBooks\* psCurrentBooksPointer) {**

**char cTempStringBuffer[100];**

**printf(**

**"Do you want to edit author's first name?: "**

**"%s(press enter if you wanna skip this step):",**

**psCurrentBooksPointer->chAuthorsFirsName);**

**gets\_s(cTempStringBuffer, 100);**

**if (strlen(cTempStringBuffer) > 2) {**

**while (nIsString(cTempStringBuffer) == 0) {**

**printf("Please reenter the string:");**

**gets\_s(cTempStringBuffer, 100);**

**}**

**strcpy(psCurrentBooksPointer->chAuthorsFirsName, cTempStringBuffer);**

**}**

**printf(**

**"Do you want to edit author's last name?: "**

**"%s(press enter if you wanna skip this step):",**

**psCurrentBooksPointer->chAuthorsLastName);**

**gets\_s(cTempStringBuffer, 100);**

**if (strlen(cTempStringBuffer) > 0) {**

**while (nIsString(cTempStringBuffer) == 0) {**

**printf("Please reenter the string:");**

**gets\_s(cTempStringBuffer, 100);**

**}**

**strcpy(psCurrentBooksPointer->chAuthorsLastName, cTempStringBuffer);**

**}**

**printf(**

**"Do you want to edit name of the book?:"**

**" %s(press enter if you wanna skip this step):",**

**psCurrentBooksPointer->chNameOfTheBook);**

**gets\_s(cTempStringBuffer, 100);**

**if (strlen(cTempStringBuffer) > 0) {**

**strcpy(psCurrentBooksPointer->chNameOfTheBook, cTempStringBuffer);**

**}**

**printf(**

**"Do you want to edit the year book was published?: %d(press enter "**

**"if you wanna skip this step):",**

**psCurrentBooksPointer->iYearPublished);**

**gets\_s(cTempStringBuffer, 100);**

**if (strlen(cTempStringBuffer) > 0) {**

**while (nIsNum(cTempStringBuffer) == 0) {**

**printf("Please reenter the number:");**

**gets\_s(cTempStringBuffer, 100);**

**}**

**psCurrentBooksPointer->iYearPublished = atoi(cTempStringBuffer);**

**}**

**printf(**

**"Do you want to edit the number of pages?: %d"**

**"(press enter if you wanna skip this step):",**

**psCurrentBooksPointer->iNumberOfPages);**

**gets\_s(cTempStringBuffer, 100);**

**if (strlen(cTempStringBuffer) > 0) {**

**while (nIsNum(cTempStringBuffer) == 0) {**

**printf("Please reenter the number:");**

**gets\_s(cTempStringBuffer, 100);**

**}**

**psCurrentBooksPointer->iNumberOfPages = atoi(cTempStringBuffer);**

**}**

**printf(**

**"Do you want to edit the price?: %.2lf(press enter if you wanna "**

**"skip this step):",**

**psCurrentBooksPointer->iPriceOfTheBook);**

**gets\_s(cTempStringBuffer, 100);**

**if (strlen(cTempStringBuffer) > 1) {**

**while (nIsRealNumber(cTempStringBuffer) == 0) {**

**printf("Please reenter the number:");**

**gets\_s(cTempStringBuffer, 100);**

**}**

**psCurrentBooksPointer->iPriceOfTheBook = atof(cTempStringBuffer);**

**}**

**}**

**//------------------------------------------------------------------------------**

**void vEditElementByIndex(struct sBooks\* pListStart) {**

**int iIndexOfEdElement = 0;**

**struct sBooks\* start = pListStart;**

**printf("Enter index of an element you want to edit:");**

**iIndexOfEdElement = iReadFoolProofInt();**

**int iCurrElementIndex = 1;**

**while (pListStart) {**

**if (iIndexOfEdElement == iCurrElementIndex) {**

**vEditElement(pListStart);**

**break;**

**}**

**iCurrElementIndex++;**

**pListStart = pListStart->pNext;**

**if (!pListStart) {**

**printf("There is no element with index %d\n", iIndexOfEdElement);**

**return;**

**}**

**}**

**printf("Do you want to resort the list: yes(1) no(0): ");**

**if (iReadFoolProofInt() == 1) {**

**vSortList(start);**

**printf("List is resorted\n");**

**}**

**printf("Element successfully edited.\n");**

**}**

**//------------------------------------------------------------------------------**

**void vReadFoolProofNameString(char testString[]) {**

**scanf("%[^\n]s", testString);**

**while (nIsString(testString) == 0) {**

**printf(**

**"The string you entered does not correspond to name string "**

**"format(no spaces or numbers). Please enter a new string:");**

**getchar();**

**scanf("%[^\n]s", testString);**

**}**

**}**

**//------------------------------------------------------------------------------**

**int iReadFoolProofInt() {**

**char buffer[60];**

**gets\_s(buffer, 59);**

**while (nIsNum(buffer) == 0) {**

**printf("Reenter the interger number: ");**

**gets\_s(buffer, 59);**

**}**

**return atoi(buffer);**

**}**

**//------------------------------------------------------------------------------**

**double dReadFoolProofDouble() {**

**char buffer[60];**

**gets\_s(buffer, 59);**

**while (nIsRealNumber(buffer) == 0) {**

**printf("Reenter the floating point number: ");**

**gets\_s(buffer, 59);**

**}**

**return atof(buffer);**

**}**

**//------------------------------------------------------------------------------**

**main.c**

**#include <stdio.h>**

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#include <windows.h>

#include "books.h"

int main() {

struct sBooks\* pCurrentBookPointer = NULL;

char cInputFileAddress[] = "input.txt";

if (readFromFile(cInputFileAddress, &pCurrentBookPointer) == 0) {

return 0;

}

HWND console = GetConsoleWindow();

RECT r;

GetWindowRect(console, &r); // stores the console's current dimensions

MoveWindow(console, r.left, r.top, 1300, 600, TRUE);

int iUserAnswer = 3;

printf("Book store management software, by Chaus Ivan \n");

while (iUserAnswer != 0) {

printf(

"\t(1)Add"

"\n\t(2)Sort List"

"\n\t(3)Print list"

"\n\t(4)Print books with price less than certain price"

"\n\t(5)Write to file"

"\n\t(6)Delete a single element"

"\n\t(7)Save "

"\n\t(8)Search by author's last name"

"\n\t(9)Print two books with the most pages "

"\n\t(10)Edit existing entry"

"\n\t(0)Exit\nYour answer:");

iUserAnswer = readFoolProofInt();

MoveWindow(console, r.left, r.top, 1000, 600, TRUE);

switch (iUserAnswer) {

case 1: {

addNewNode(&pCurrentBookPointer);

break;

}

case 2: {

sortList(pCurrentBookPointer);

printf("\nList successfully sorted by author's last name\n\n");

break;

}

case 3: {

printList(pCurrentBookPointer);

break;

}

case 4: {

lessThanCertainPrice(&pCurrentBookPointer);

break;

}

case 5: {

writeToFile(pCurrentBookPointer);

break;

}

case 6: {

deleteSingleElement(&pCurrentBookPointer);

break;

}

case 7: {

saveChanges(pCurrentBookPointer);

break;

}

case 8: {

searchByAuthorsLastName(&pCurrentBookPointer);

break;

}

case 9:

booksWithTheMostPages(pCurrentBookPointer);

break;

case 10: {

editElementByIndex(pCurrentBookPointer);

break;

}

case 0: {

return 0;

break;

}

default: {

printf("\nYou have entered the wrong number. Please try again\n");

}

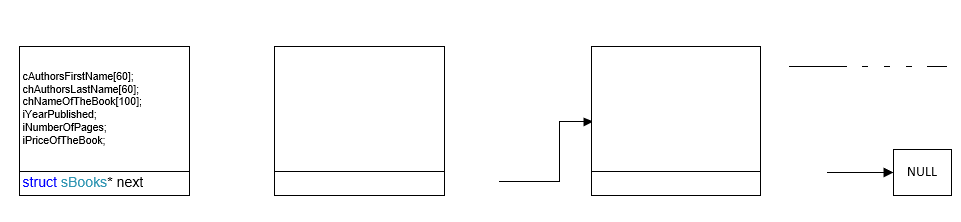
}

}

**}**

**Частина 3.**

Зразки фігур, які були використані для схематичного зображення структур.



**Висновок**

На даній лабораторній роботі я детальніше вивчив етап проектування та етап кодування ЖЦ ПЗ. Також я задокументував дані етапу згідно заданому стандарту і створив необхідний пакет документів, у який входять схематичне зображення використаних структур даних, блок-схеми основних алгоритмів програми, код програми та користувацька бібліотека зразків Visio-фігур, які були створенні для схематичного зображення структур даних.

