Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**Звіт**

Про виконання лабораторної роботи №2

**На тему:**

«Документування етапів проектування та кодування програми»

з дисципліни “Вступ до інженерії програмного забезпечення”

**Лектор:**

ст. вик. каф. ПЗ

Левус Є.В.

**Виконав:**

ст. гр. ПЗ-18

Юшкевич А.І.

**Прийняв:**

асис. каф. ПЗ

Вишневський О.В.

« … » … 2023 р.

∑ = \_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2023

**Тема.** Документування етапів проектування та кодування програми.

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

**Теоретичні відомості**

**2.** У чому полягає етап проектування для найпростішої програми?

Обрання мови програмування, опис компонентів програми, визначення структури даних (за потреби)

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

Ім’я класу записується з префіксом “C” на початку, кожне слово пишеться з великої літери: “CMyClass”, “СHuman”.

Відповідно до стандарту Microsoft секції керування доступом розташовуються в такій послідовності: public – protected – private.

Послідовність розташування членів класу в кожній з секцій private, protected та public:

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

**33.** Перерахуйте найвикористовуваніші мови програмування.

1) Python

2) C

3) C++

4) C#

5) Java

\*According to IEEE Spectrum’s ranking

**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 запропонувати алгоритми зчитування та запису у файл, сортування, пошук, редагування, видалення елементів та решта функцій згідно з варіанту.

**Частина ІІ.** Сформувати пакет документів до розробленої раніше власної програми:

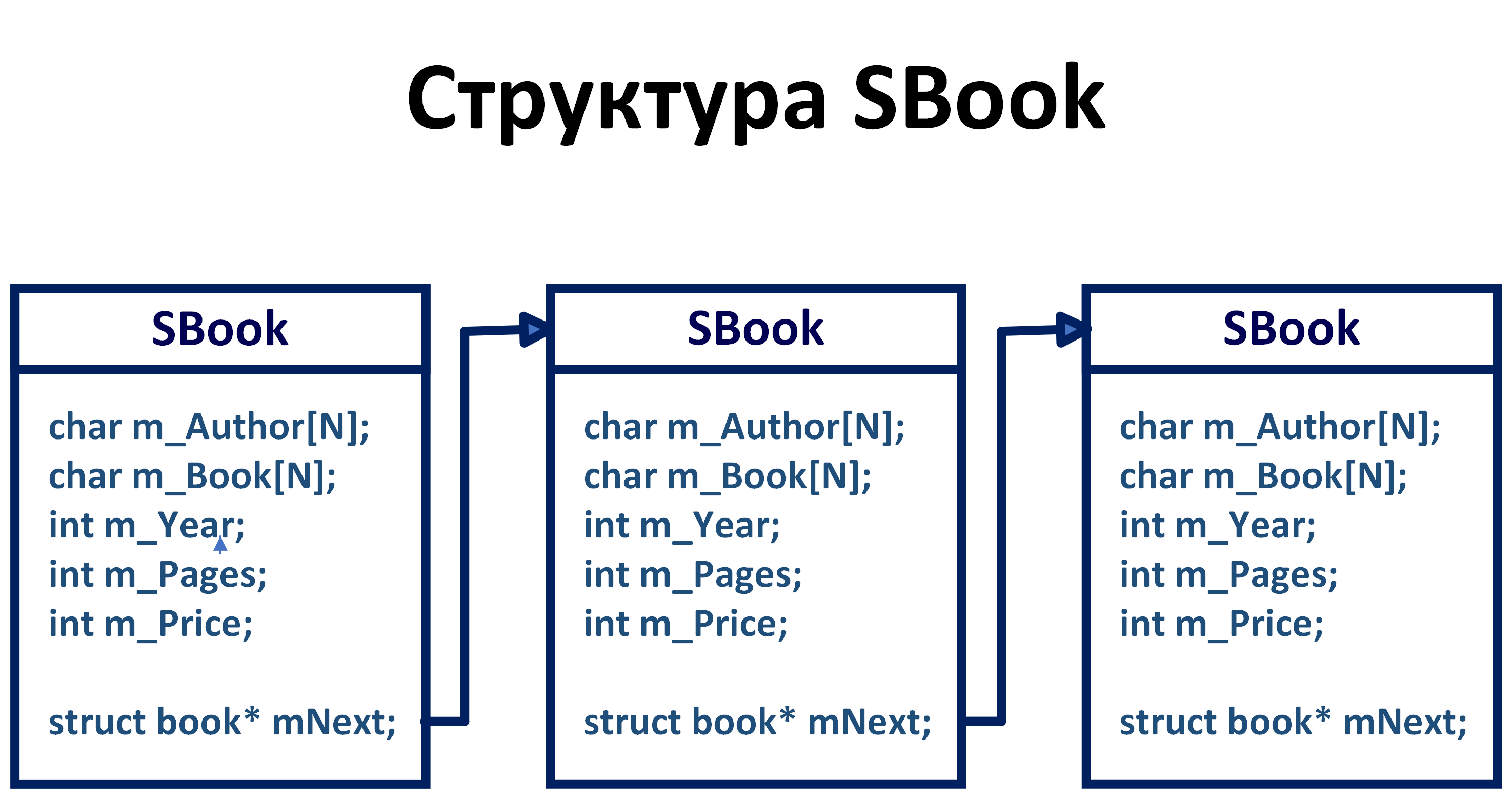
1. Схематичне зображення структур даних, які використовуються для збереження інформації;
2. Блох-схема алгоритмів – основної функції й двох окремих функцій-підпрограм (наприклад, сортування та редагування);
3. Текст програми з коментарями та оформлений згідно вище наведених рекомендацій щодо забезпечення читабельності й зрозумілості.

Для схематичного зображення структур даних, блок-схеми алгоритму використати редактор MS-Visio.

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

**Хід роботи**

Розроблені блок-схеми для всіх функцій та структури:



**main()**

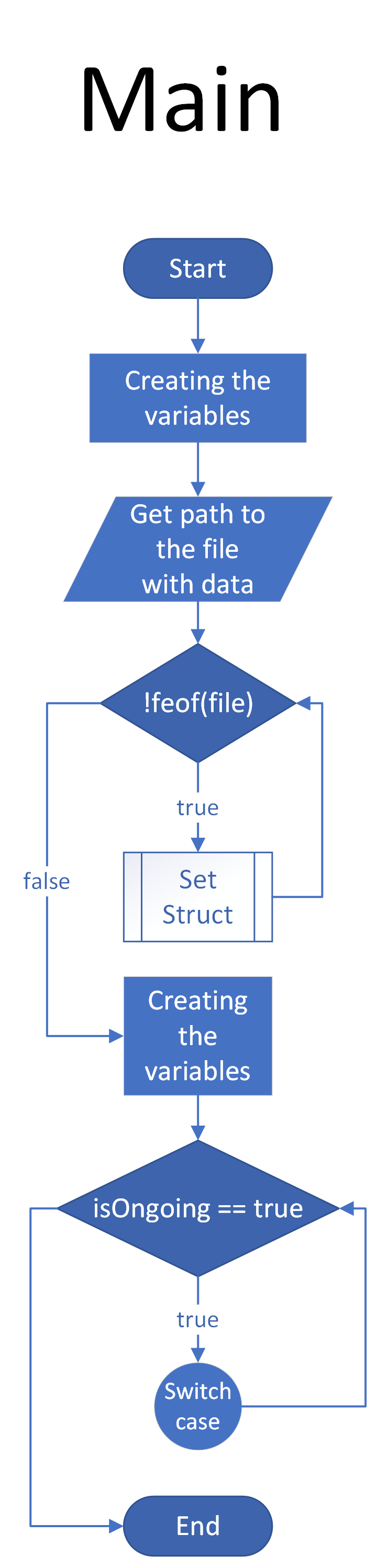
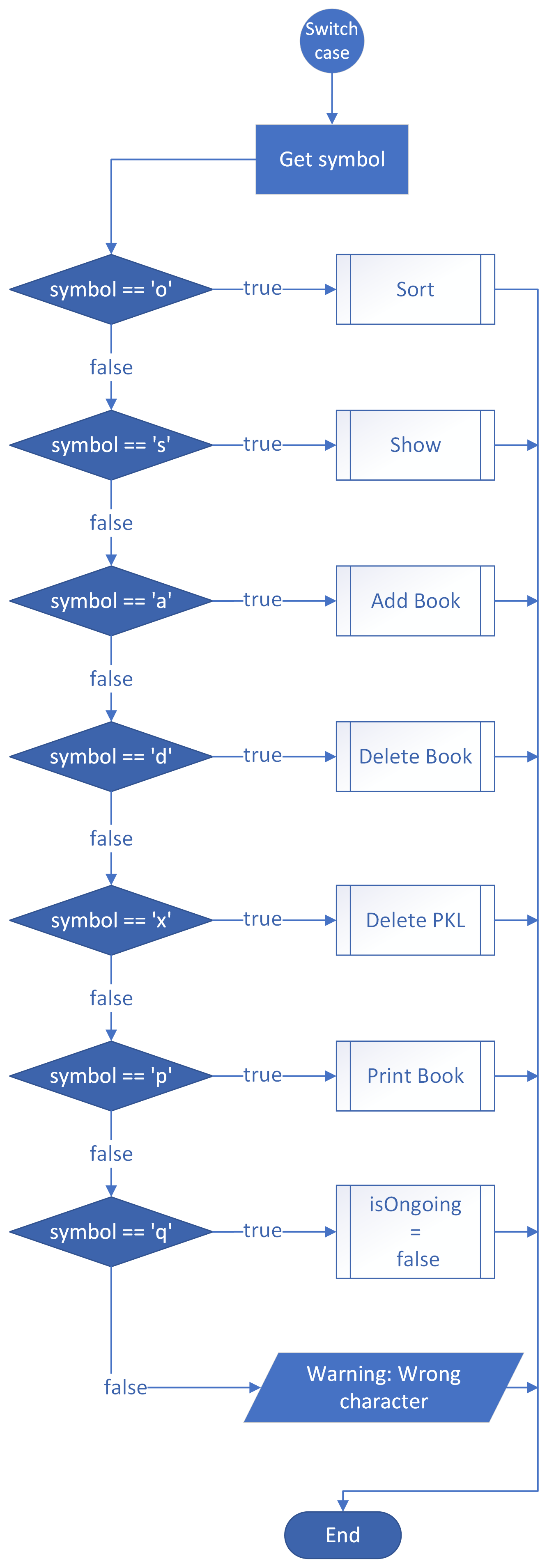


Рис. 1. Блок-схема алгоритму роботи функції main()

**DeletePKL()**

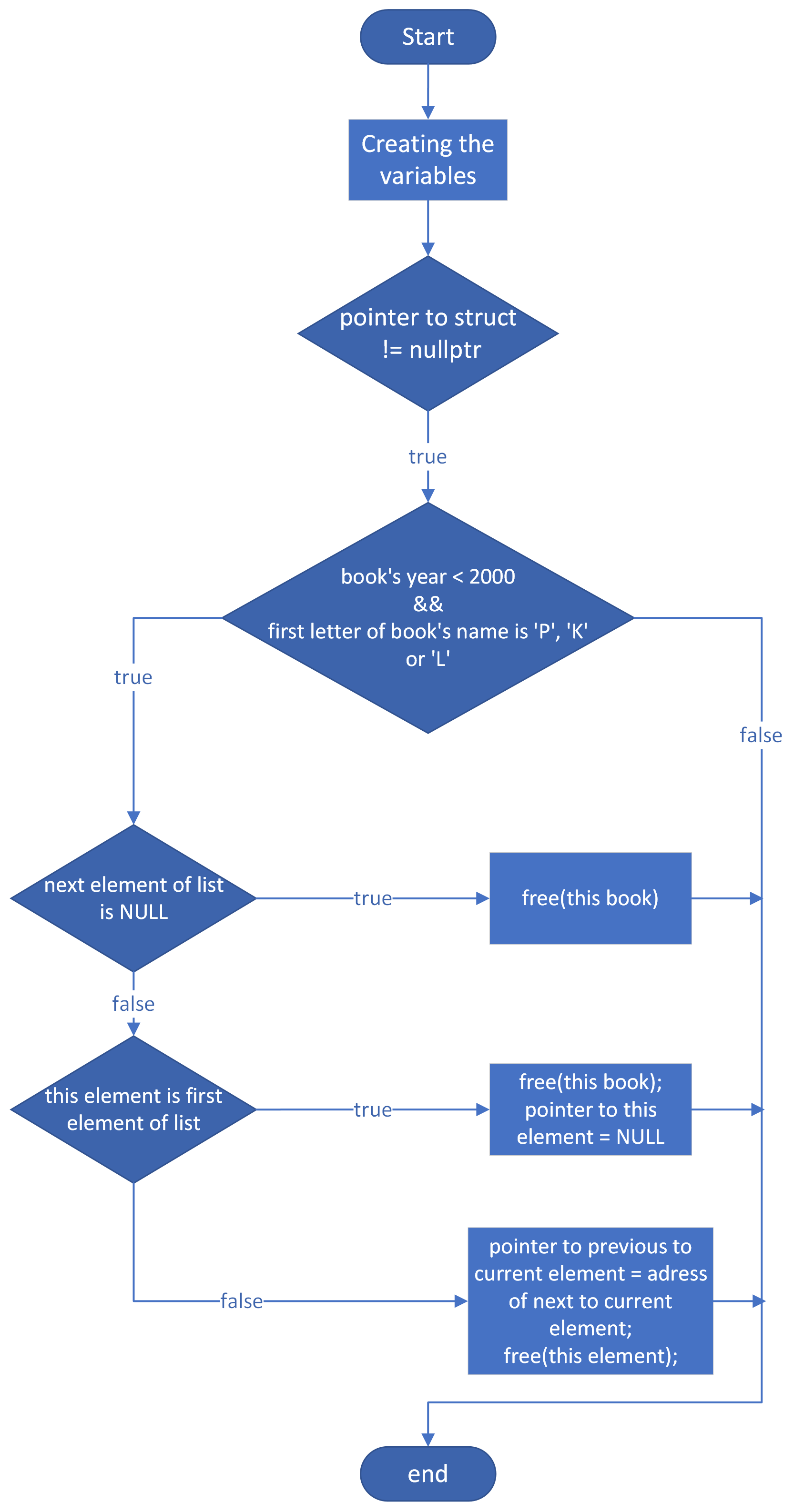


Рис. 2. Блок-схема алгоритму роботи функції DeletePKL()

**SetStruct()**

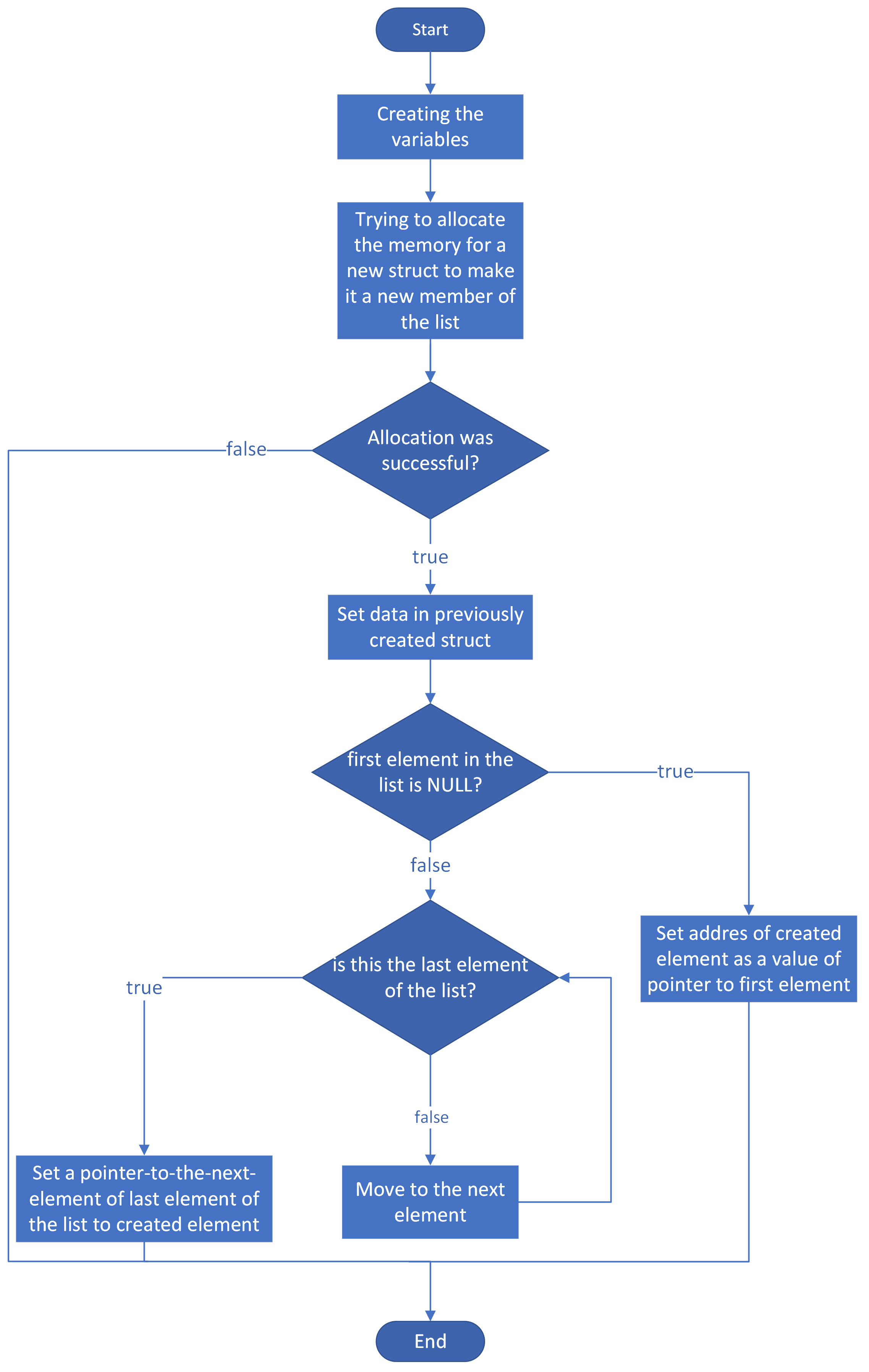


Рис. 3. Блок-схема алгоритму роботи функції SetStruct()

**Код програми**

**Lab\_02\_SE.cpp:**

#include "Header.h"

int main() {

char cSymbol;

char sPath[STR];

char sTempStr[STR];

SBook\* p\_sbFirstBook = NULL;

SBook\* p\_sbFree = NULL;

cout << "Enter path of file with list: ";

cin >> sPath;

FILE\* fFile = fopen(sPath, "r");

while (!feof(fFile)) {

fgets(sTempStr, STR, fFile);

SetStruct(&p\_sbFirstBook, sTempStr);

}

fclose(fFile);

int nElementIndex = 0;

int bIsOngoing = 1;

getchar();

while (bIsOngoing) {

cout << "\n\nKeys: q - Quit, s - Show, o - Sort, a - Add, d - Delete, x - Delete books which names start with П К or Л\n\nPlease, choose option: ";

cSymbol = getchar();

switch (cSymbol){

case 'o':

Sort(&p\_sbFirstBook);

break;

case 's':

Show(p\_sbFirstBook);

break;

case 'a':

AddElement(&p\_sbFirstBook);

break;

case 'd':

cout << "\nEnter number of element: ";

cin >> nElementIndex;

DeleteElement(&p\_sbFirstBook, nElementIndex);

break;

case 'x':

DeletePKL(&p\_sbFirstBook);

break;

case 'p':

Print(p\_sbFirstBook);

break;

case 'q':

bIsOngoing = 0;

if (p\_sbFirstBook) {

SBook\* p\_sbElement = p\_sbFirstBook;

while (p\_sbElement) {

p\_sbFree = p\_sbElement;

p\_sbElement = p\_sbElement->mNext;

free(p\_sbFree);

}

p\_sbFree = NULL;

p\_sbFirstBook = NULL;

}

break;

default:

cout << "Wrong character!";

break;

}

getchar();

}

return 0;

}

**Header.h:**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <string>

#include <string.h>

#include <ctype.h>

#include <Windows.h>

using namespace std;

#define N 100

#define STR 200

typedef struct book {

char mAuthor[N];

char mBook[N];

int mYear;

int mPages;

int mPrice;

struct book\* mNext;

} SBook;

void SetStruct(SBook\*\* p\_sbFirstBook, char\* p\_cString);

void Show(SBook\* p\_sbFirstBook);

void Print(SBook\* p\_sbFirstBook);

void Sort(SBook\*\* p\_sbFirstBook);

SBook\* GetConcrete(SBook\* p\_sbFirstBook, int nIndex);

void DeleteElement(SBook\*\* p\_sbFirstBook, int nIndex);

void AddElement(SBook\*\* p\_sbFirstBookS);

void DeletePKL(SBook\*\* p\_sbFirstBook);

SBook\* GetLast(SBook\* p\_sbFirstBook);

**Class\_Functions.cpp:**

#include "Header.h"

//The SetStruct function takes in a double pointer to an SBook structure and a pointer to a string.

//The function parses the string using strtok and assigns the values to the fields of the dynamically allocated SBook structure.

void SetStruct(SBook\*\* pp\_sbFirstBook, char\* pString) {

char\* pToken = NULL;

SBook\* p\_sbTempBook = new SBook;

if (!p\_sbTempBook) {

cout << "\n\nSorry, can't allocate the memory\n\n";

}

else {

pToken = strtok(pString, "\t");

strcpy(p\_sbTempBook->mAuthor, pToken);

pToken = strtok(NULL, "\t");

strcpy(p\_sbTempBook->mBook, pToken);

pToken = strtok(NULL, "\t");

p\_sbTempBook->mYear = atoi(pToken);

pToken = strtok(NULL, "\t");

p\_sbTempBook->mPages = atoi(pToken);

pToken = strtok(NULL, "\t");

p\_sbTempBook->mPrice = atoi(pToken);

p\_sbTempBook->mNext = nullptr;

if (!\*pp\_sbFirstBook) {

\*pp\_sbFirstBook = p\_sbTempBook;

}

else {

SBook\* p\_sbRunner = \*pp\_sbFirstBook;

while (p\_sbRunner->mNext) {

p\_sbRunner = p\_sbRunner->mNext;

}

p\_sbRunner->mNext = p\_sbTempBook;

}

}

}

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

// This function displays a linked list of SBook structures.

void Show(SBook\* p\_sbFirstBook) {

cout << "\n\n\tAuthor\t\t\tBook\t\t\t\t\tYear\tPages\tPrice\n\n\n";

for (; p\_sbFirstBook; p\_sbFirstBook = p\_sbFirstBook->mNext) {

printf( "%-20s \t%-35s\t\t%d\t%d\t%d\n\n", p\_sbFirstBook->mAuthor, p\_sbFirstBook->mBook, p\_sbFirstBook->mYear,

p\_sbFirstBook->mPages, p\_sbFirstBook->mPrice);

}

}

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

// This function sorts a linked list of SBook structures in descending order

// based on the price of the book and alphabetical order of book names.

void Sort(SBook\*\* pp\_sbFirstBook) {

SBook\* p\_sbElement = \*pp\_sbFirstBook;

SBook sbTempBook;

SBook\* p\_sbBooksMinusOne = NULL;

int nAveragePrice = 0, nCounter = 0;

// Calculate the average price of books

for (p\_sbElement = \*pp\_sbFirstBook; p\_sbElement->mNext; p\_sbElement = p\_sbElement->mNext) {

nAveragePrice += p\_sbElement->mPrice;

nCounter++;

}

nAveragePrice /= nCounter;

// Perform bubble sort on linked list

for (int i = 0; i < nCounter - 1; i++) {

for (int j = nCounter - 1; j > i + 1; j--) {

p\_sbElement = GetConcrete(\*pp\_sbFirstBook, j);

p\_sbBooksMinusOne = GetConcrete(\*pp\_sbFirstBook, j - 1);

if (p\_sbElement->mPrice > nAveragePrice && strcmp(p\_sbElement->mBook, p\_sbBooksMinusOne->mBook) < 0) {

sbTempBook = \*p\_sbBooksMinusOne;

\*p\_sbBooksMinusOne = \*p\_sbElement;

p\_sbBooksMinusOne->mNext = sbTempBook.mNext;

sbTempBook.mNext = p\_sbElement->mNext;

\*p\_sbElement = sbTempBook;

}

}

}

cout << "\n\nList was successfully sorted!\n\n";

}

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

// This function returns a pointer to the book with index "index" in the list

// "SFirstBook" of books

SBook\* GetConcrete(SBook\* sbFirstBook, int nIndex) {

SBook\* p\_sbElement = NULL;

if (nIndex > 0)

p\_sbElement = sbFirstBook;

while (--nIndex > 0) {

p\_sbElement = p\_sbElement->mNext;

}

return p\_sbElement;

}

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

//Deletes an element from the linked list of SBook.

void DeleteElement(SBook\*\* pp\_sbFirstBook, int nIndex) {

// Declare and initialize variables

SBook\* p\_sbElement = \*pp\_sbFirstBook;

SBook\* p\_sbPrevBook = p\_sbElement;

// Check if index is valid

if (nIndex <= 0) {

cout << "WARNING: You entered the wrong index";

return;

}

// Traverse the linked list to find the element to delete

while (--nIndex) {

if (p\_sbElement->mNext == NULL) {

cout << "WARNING: You entered the wrong index";

return;

}

p\_sbPrevBook = p\_sbElement;

p\_sbElement = p\_sbElement->mNext;

}

// Delete the element

if (p\_sbElement->mNext == NULL) {

p\_sbPrevBook->mNext = NULL;

free(p\_sbElement);

p\_sbElement = NULL;

}

else if (p\_sbElement == \*pp\_sbFirstBook) {

\*pp\_sbFirstBook = p\_sbElement->mNext;

free(p\_sbElement);

p\_sbElement = NULL;

}

else {

p\_sbPrevBook->mNext = p\_sbElement->mNext;

free(p\_sbElement);

p\_sbElement = NULL;

}

cout << "\n\nElement was successfully deleted!\n\n";

}

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

// This function adds a new element to the linked list of SBook structs

void AddElement(SBook\*\* pp\_sbFirstBook) {

SBook\* p\_sbNewElement = new SBook;

SBook\* p\_sbElement = \*pp\_sbFirstBook;

SBook\* p\_sbPrevBook = nullptr;

int iIndex = 0;

bool bBadIndex = false;

getchar();

cout << "\nPlease, enter author's name: ";

gets\_s(p\_sbNewElement->mAuthor);

cout << "\nPlease, enter book's name: ";

gets\_s(p\_sbNewElement->mBook);

cout << "\nPlease, enter book's year: ";

cin >> p\_sbNewElement->mYear;

cout << "\nPlease, enter number of pages: ";

cin >> p\_sbNewElement->mPages;

cout << "\nPlease, enter book's price: ";

cin >> p\_sbNewElement->mPrice;

cout << "\nPlease, enter index where I should paste your element: ";

cin >> iIndex;

if (iIndex <= 0)

bBadIndex = true;

while (--iIndex > 0 && !bBadIndex) {

p\_sbPrevBook = p\_sbElement;

p\_sbElement = p\_sbElement->mNext;

}

if (!pp\_sbFirstBook && iIndex)

bBadIndex = true;

if (bBadIndex) {

cout << "WARNING: You entered the wrong index";

return;

}

if (p\_sbElement == nullptr) {

\*pp\_sbFirstBook = p\_sbNewElement;

}

else if (p\_sbElement == \*pp\_sbFirstBook) {

p\_sbNewElement->mNext = p\_sbElement;

\*pp\_sbFirstBook = p\_sbNewElement;

}

else {

p\_sbNewElement->mNext = p\_sbElement;

p\_sbPrevBook->mNext = p\_sbNewElement;

}

cout << "\n\nElement was successfully added!\n\n";

}

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

// This function deletes all the books from the linked list whose year is less than 2000 and whose book name starts with "P", "K" or "L"

void DeletePKL(SBook\*\* p\_sbFirstBook) {

const int N\_YEAR{ 2000 };

SBook\* sbElement = \*p\_sbFirstBook;

SBook\* sbPrevBook = \*p\_sbFirstBook;

char p[3] = "P";

char k[3] = "K";

char l[3] = "L";

while (sbElement) {

if (sbElement->mYear < N\_YEAR && !(strncmp(sbElement->mBook, p, 1) && strncmp(sbElement->mBook, k, 1) && strncmp(sbElement->mBook, l, 1))) {

if (sbElement->mNext == NULL) {

sbPrevBook->mNext = sbElement->mNext;

free(sbElement);

sbElement = NULL;

continue;

}

else if (sbElement == \*p\_sbFirstBook) {

\*p\_sbFirstBook = sbElement->mNext;

free(sbElement);

sbElement = \*p\_sbFirstBook;

continue;

}

else {

sbPrevBook->mNext = sbElement->mNext;

free(sbElement);

sbElement = sbPrevBook->mNext;

continue;

}

}

sbPrevBook = sbElement;

sbElement = sbElement->mNext;

}

cout << "\n\nElements were successfully deleted!\n\n";

}

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

//This function returns a pointer to the last element of a singly - linked list of SBook nodes

SBook\* GetLast(SBook\* p\_sbFirstBook) {

SBook\* p\_sbElement = p\_sbFirstBook;

while (p\_sbElement->mNext) {

p\_sbElement = p\_sbElement->mNext;

}

return p\_sbElement;

}

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

// This function is used to print the linked list of SBooks into a file with the provided path.

void Print(SBook\* p\_sbFirstBook) {

char path[STR];

cout << "\n\nEnter path: ";

cin >> path;

FILE\* fFile = fopen(path, "w");

for (; p\_sbFirstBook; p\_sbFirstBook = p\_sbFirstBook->mNext) {

fprintf(fFile, "%-20s \t%-35s\t\t%d\t%d\t%d\n\n", p\_sbFirstBook->mAuthor, p\_sbFirstBook->mBook, p\_sbFirstBook->mYear,

p\_sbFirstBook->mPages, p\_sbFirstBook->mPrice);

}

cout << "\n\nElement was saved successfully!\n\n";

}

**Висновок**

Під час виконання даної лабораторної роботи я покращив свої знання в інженерії програмного забезпечення, завдяки ним, я зміг оптимізувати свою розробку коду та виправив свої потенційні помилки(в коді) ще на етапі кодування, також засвоїв початкові навички документування своєї роботи та розробив декілька блок-схем для пояснення свого коду, а також використовував для коментарі для ще точнішого пояснення. Також, використав угорську нотацію для спрощення читання мого коду іншим.