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
«Документування етапів проектування та кодування програми»
з дисципліни «Вступ до інженерії програмного забезпечення»

Лектор:
доцент кафедри ПЗ
Левус Є.В.

Виконала:
студ. групи ПЗ-15
Мартиняк А. В.

Прийняв:
асистент кафедри ПЗ
Самбір А. А.

«___» _____ 2021 р.
∑ = ____ _____

Львів – 2022

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

Теоретичні відомості

5. Для якого етапу ЖЦ ПЗ проектні рішення є вхідними? Наведіть приклади проектних рішень.

Проектні рішення є вхідними даними для етапу кодування. Проектні рішення можуть бути такими: блок-схема, діаграма, макет і т.д.

30. Як можна провести навігацію по коду?

ReSharper C ++ дозволяє дуже швидко здійснювати навігацію по великих проектах і шукати в них потрібний код. Ось деякі з можливостей навігації:

- ✓ Go to Everything знаходить за назвою будь-який тип сутності у всьому проекті відразу. Результати також можна фільтрувати підкомандами, такими як Go to Type, Go to File і Go to Symbol.
- ✓ Go to File Member дозволяє швидко знайти символ в тому файлі, де ви зараз працюєте.
- ✓ Go to Base / Derived допомагає шукати нащадків і батьків класів або елементів класу. Go to Definition дозволяє швидко перейти до місця оголошення певного об'єкту.
- ✓ Go to Related Files застосовується для переходу до файлів, які мають якесь відношення до поточного. Це можуть бути включені заголовки або cpp-файли. Крім того, ReSharper дає змогу отримати доступ між заголовним і відповідним cpp-файлом (Ctrl + B).

35. Перерахуйте найвикористовуваніші мови програмування.

Python, C, Java, C++, C#, Visual Basic, JavaScript, Assembly language, SQL, PHP.

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 функцій-підпрограм (наприклад, сортування та редагування);
3. Текст програми з коментарями та оформлений згідно вище наведених рекомендацій щодо забезпечення читабельності й зрозумілості.

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

Умова програми

З текстового файлу зчитати послідовність записів, які містять дані про книгу: Автор, Назва книги, Рік видання, Кількість сторінок, Вартість. Роздрукувати введені дані у вигляді таблиці, а також подати інформацію згідно варіанту.

Відсортувати за назвою в алфавітному порядку дані про книги, вартість яких більша середньої в бібліотеці. Реалізувати операцію вставки нового елемента у відсортований список і операцію вилучення зі списку даних, які відповідають одній з наступних умов: про книги видані раніше 2000 року з кількістю сторінок меншою 150.

Отримані результати

1. Схематичне зображення структур даних, які використовуються для збереження інформації (рис. 2.1).

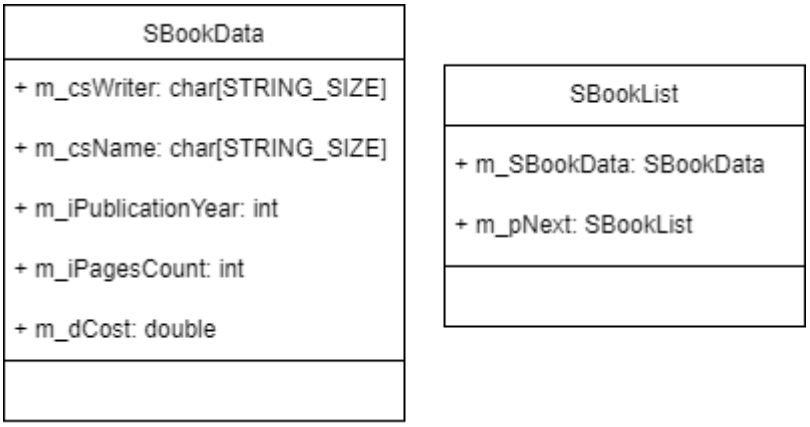

Рис. 2.1

2. Блок-схеми алгоритмів – основної функції та двох-підпрограм: main (рис. 2.2), SortByBookName (рис. 2.3), DeleteBooksByYearAndPages (рис. 2.4).

Рис. 2.2

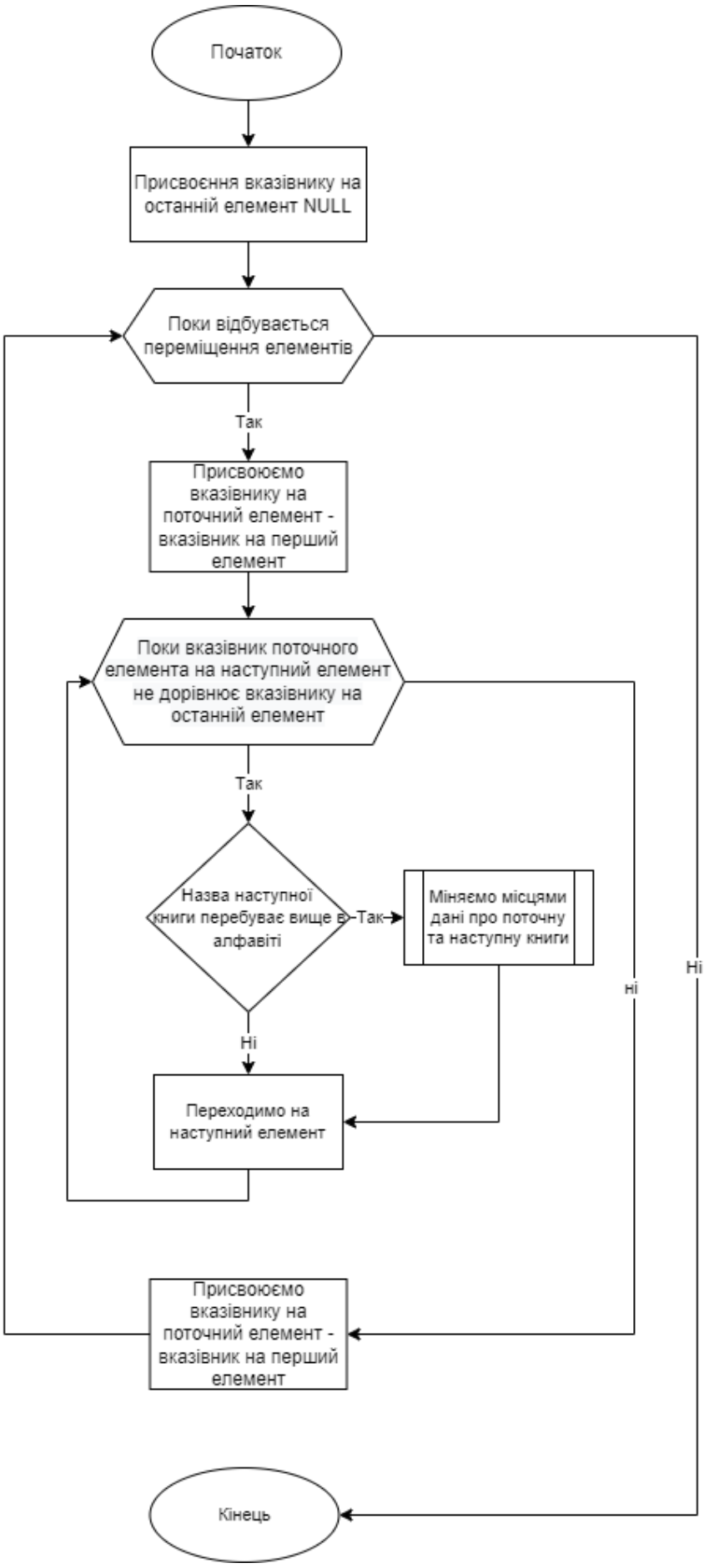
Початок
Присвоєння вказівнику на останній елемент NULL
Поки відбувається переміщення елементів
Так
Присвоюємо вказівнику на поточний елемент - вказівник на перший елемент
Поки вказівник поточного елемента на наступний елемент не дорівнює вказівнику на останній елемент
Так
Назва наступної книги перебуває вище в алфавіті
Так
Міняємо місцями дані про поточну та наступну книги
Ні
Переходимо на наступний елемент
ні
Ні
Присвоюємо вказівнику на поточний елемент - вказівник на перший елемент
Кінець

Рис. 2.3

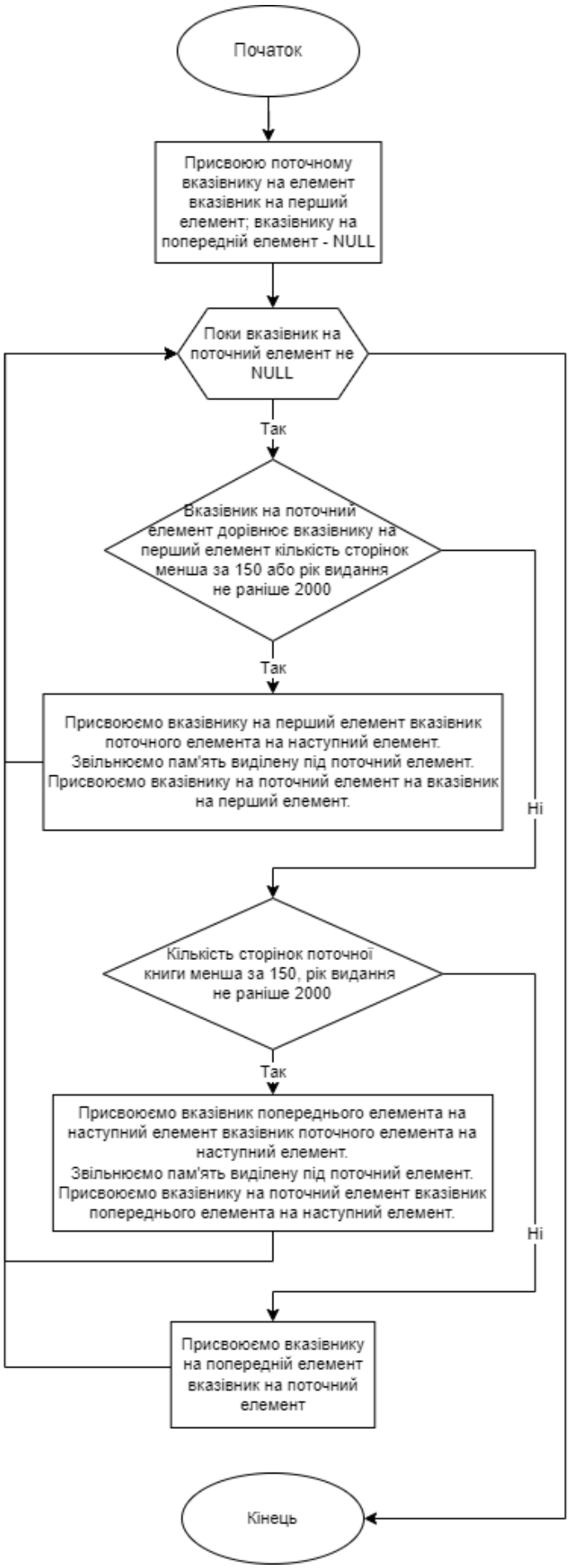
Початок
Присвоюю поточному вказівнику на елемент вказівник на перший елемент; вказівнику на попередній елемент - NULL
Поки вказівник на поточний елемент не NULL
Так
Вказівник на поточний елемент дорівнює вказівнику на перший елемент кількість сторінок менша за 150 або рік видання не раніше 2000
Так
Присвоюємо вказівнику на перший елемент вказівник поточного елемента на наступний елемент.
Звільнюємо пам'ять виділену під поточний елемент.
Присвоюємо вказівнику на поточний елемент на вказівник на перший елемент.
Ні
Кількість сторінок поточної книги менша за 150, рік видання не раніше 2000
Так
Присвоюємо вказівник попереднього елемента на наступний елемент вказівник поточного елемента на наступний елемент.
Звільнюємо пам'ять виділену під поточний елемент.
Присвоюємо вказівнику на поточний елемент вказівник попереднього елемента на наступний елемент.
Ні
Присвоюємо вказівнику на попередній елемент вказівник на поточний елемент
Кінець

Рис. 2.4

3. Текст програми з коментарями та оформлений згідно з наведеними рекомендаціями щодо забезпечення читабельності й зрозумілості.

„Main.c”

```
#include "header.h"

int main()
{
    SBookList* pLast, *pHead;

    //отримую дані з файлу
    pHead = GetDataFromFile(&pLast);

    printf("\n\t\t\t\t\tInitial list of books:\n");
    PrintList(pHead);

    // видаляю книги, вартість яких нижча середньої
    pHead = DeleteBooksByAvCost(pHead, &pLast);

    //сортую список
    SortByBookName(pHead);
    printf("\n\n\t\t\t\t\tSorted list of books:\n");
    PrintList(pHead);

    //отримую дані про нові книги
    GetNewBooksData(pLast);

    // видаляю дані про книги, які видані раніше 2000 року
    // чи кількість сторінок яких менша 150-ти
    pHead = DeleteBooksByYearAndPages(pHead);

    //сортую за назвою
    SortByBookName(pHead);

    printf("\n\n\t\tList of books with more than 150 pages and published not earlier than
2000\n");
    PrintList(pHead);

    FreeList(pHead);

    return 0;
}
```

„Header.h”

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

// максимальна довжина стрічки даних про книги
#define STRUCT_SIZE 500

// кількість символів '-' для друкування таблиці
#define COUNT_OF_LINES 52

// максимальна довжина для полів Автор та назви
#define STRING_SIZE 50
```

```c
/*Структура Дані Книги, що містить наступні поля: прізвище і ім'я автора,
* назву книги, рік видання, кількість сторінок та вартість.
* Структура Книга, що містить наступні поля: дані про книгу, що вкладені
* в структуру Дані Книги та вказівник на наступний елемент типу Список Книг.
* Приклад створення об'єкту структури:
* SBook Book1;
* Book1->m_SBookData.m_csTitle = "Koran";
*/

typedef struct SBookData {
    char m_csAuthor[STRING_SIZE];
    char m_csName[STRING_SIZE];
    int m_nPublicationYear;
    int m_nPagesCount;
    double m_dCost;
} SBookData;

typedef struct SBookList {
    SBookData m_SBookData;
    struct SBookList* m_pNext;
} SBookList;

SBookList* GetDataFromFile(SBookList** pLast);
void GetNewBooksData(SBookList* pLast);
void SwapBooksData(SBookList* pSwap1, SBookList* pSwap2);
SBookList* DeleteBooksByAvCost(SBookList* pHead, SBookList** pLast);
void SortByBookName(SBookList* pHead);
void PrintLines(int count);
void PrintList(SBookList* p);
SBookList* AddNewElement(SBookList** a);
SBookList* DeleteBooksByYearAndPages(SBookList* pHead);
void FreeList(SBookList* pHead);

#endif
```

„Functions.c”

```c
#include "header.h"

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

/*
 * Функція зчитування даних про книги приймає вказівник на вказівник на
 * останній елемент списку та повертає вказівник на початковий елемент
 * списку. Функція перевіряє, чи файл існує, якщо ні - у консоль виводиться
 * відповідне повідомлення, якщо так - відкривається файл для зчитування даних.
 * Дані про книги у файлі мають бути подані так:
 * <Автор>, <Назва>, <Рік видання>, <Кількість сторінок>, <Вартість>
 *
 * Вказівник ділить стрічку на лексеми за комою. За допомогою вказівника
 * рухаємось по стрічці даних файлу та присвоюємо елементам структури дані,
 * приведені до необхідного типу (стрічка, стрічка, ціле число, ціле число,
 * дійсне число).
 * Під кожен наступний елемент функція виділяє пам'ять динамічно, тому після
 * використання списку книг пам'ять необхідно очистити.
 */

SBookList* GetDataFromFile(SBookList** pLast) {
    FILE* pfBook;
```

```
    pfBook = fopen("booksData.txt", "rt");
    if (!pfBook){
        printf("The file \"books.txt\" cannot be opened");
        exit(1);
    }

    (*pLast) = (SBookList*)malloc(sizeof(SBookList));
    SBookList* pHead = *pLast, *pNewBook = pHead, *pCurrBook = NULL;
    char* pcReadStr, scDataStr[STRUCT_SIZE];

    //зчитування даних з файлу
    while (!feof(pfBook)){
        pCurrBook = pNewBook;

        fgets(scDataStr, STRUCT_SIZE, pfBook);

        pcReadStr = (char*)strtok(scDataStr, ",");
        strcpy(pCurrBook->m_SBookData.m_csAuthor, pcReadStr);

        pcReadStr = (char*)strtok(NULL, ",");
        strcpy(pCurrBook->m_SBookData.m_csName, pcReadStr);

        pcReadStr = strtok(NULL, ",");
        pCurrBook->m_SBookData.m_nPublicationYear = atoi(pcReadStr);

        pcReadStr = strtok(NULL, ",");
        pCurrBook->m_SBookData.m_nPagesCount = atoi(pcReadStr);

        pcReadStr = strtok(NULL, ",");
        pCurrBook->m_SBookData.m_dCost = atof(pcReadStr);

        pCurrBook->m_pNext = (SBookList*)malloc(sizeof(SBookList));
        pNewBook = pCurrBook->m_pNext;
    }

    //занулювання вказівника pNext останнього елемента
    free(pCurrBook->m_pNext);
    pCurrBook->m_pNext = NULL;
    (*pLast) = pCurrBook;

    fclose(pfBook);

    return pHead;
}

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

//Функція для обміну місцями книг у списку.
/*
* Приймає вказівники на елементи, які потрібно переставити місцями; не повертає
* нічого. Пам'ять під тимчасовий вказівник виділяється та звільнюється
* автоматично.
```

```
*/
void SwapBooksData(SBookList* pSw1, SBookList* pSw2) {
    SBookList* pTemp = (SBookList*)malloc(sizeof(SBookData));
    pTemp->m_SBookData = pSw1->m_SBookData;
    pSw1->m_SBookData = pSw2->m_SBookData;
    pSw2->m_SBookData = pTemp->m_SBookData;
    free(pTemp);
}

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

// Функція сортування списку за алфавітом по назві книги
/*
* Приймає значення вказівника на перший елемент списку, не повертає нічого.
* Відбувається сортування книг за назвою по алфавіту в зростаючому порядку.
* Перестановка книг місцями відбувається за допомогою функції SwapBooksData.
*/

void SortByBookName(SBookList* pHead) {
    if (!pHead) return;

    int bSwapped;
    SBookList* pCurrBook, * pLastBook = NULL;

    if (pHead == NULL) return;

    do {
        bSwapped = false;
        pCurrBook = pHead;

        while (pCurrBook->m_pNext != pLastBook) {
            if (strcmp(pCurrBook->m_pNext->m_SBookData.m_csName,
                       pCurrBook->m_SBookData.m_csName) < 0) {
                SwapBooksData(pCurrBook, pCurrBook->m_pNext);
                bSwapped = true;
            }
            //перехід на наступний елемент
            pCurrBook = pCurrBook->m_pNext;
        }
        //"точка зупинки" сортування
        pLastBook = pCurrBook;
    } while (bSwapped);
}

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

// Функція для отримання та додавання довільної кількості нових книг у кінець
// списку
/*
* Приймає  вказівник на останній елемент списку, не повертає нічого.
* Для отримання даних про нові книги викликається функція AddNewElement.
*/
```

```
void GetNewBooksData(SBookList* pLast) {
    int nAddCount;
    printf("\n\nHow many elements do you want to add?  ");
    scanf("%d", &nAddCount);
    int nLeftCount = nAddCount;

    //поки не введено дані про всі книги
    while (nLeftCount) {
        if (nAddCount == nLeftCount){
            printf("\n\nPlease enter information about the first book.\n");
        } else {
            printf("\n\nPlease enter information about the next book.\n");
        }
        pLast = AddNewElement(&pLast);
        --nLeftCount;
    }
}

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

// Функція отримання даних про книгу, яку хоче додати користувач
/*
* Приймає вказівник на вказівник на останній елемент у списку та повертає
* вказівник на новий останній елемент списку. Пам'ять під нові елементи
* виділяється динамічно, тому після використання списку книг пам'ять необхідно
* очистити.
*/

SBookList* AddNewElement(SBookList** ppNewBook) {
    char csAuthor[50]; char csName[50]; int nPublicationYear; int nPagesCount; double dCost;

    //отримання даних
    printf("Author's name: ");
    scanf(" %[^\n]s", csAuthor);
    printf("Book title: ");
    scanf(" %[^\n]s", csName);
    printf("Year of publication of the book: ");
    scanf("%d", &nPublicationYear);
    printf("The number of pages: ");
    scanf("%d", &nPagesCount);
    printf("Cost of the book: ");
    scanf("%lf", &dCost);

    SBookList* temp = (SBookList*)malloc(sizeof(SBookList));
    temp->m_pNext = NULL;
    (*ppNewBook)->m_pNext = temp;

    //копіювання даних в список книг
    strcpy((*ppNewBook)->m_SBookData.m_csAuthor, csAuthor);
    strcpy((*ppNewBook)->m_SBookData.m_csName, csName);
    (*ppNewBook)->m_SBookData.m_nPublicationYear = nPublicationYear;
```

```
    (*ppNewBook)->m_SBookData.m_nPagesCount = nPagesCount;
    (*ppNewBook)->m_SBookData.m_dCost = dCost;

    (*ppNewBook)->m_pNext = NULL;
    return (*ppNewBook);
}

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

// Функція видалення зі списку книг, за ознакою
/*
* Функція приймає вказівник на перший елемент списку та повертає новий
* вказівник на перший елемент списку. Функція видаляє книги, що були
* опубліковані раніше 2000 року та кількість сторінок яких менша за 150.
* Пам'ять видалених елементів очищується автоматично.
*/
SBookList* DeleteBooksByYearAndPages(SBookList *pHead) {
    if (!pHead) exit(0);
    SBookList* pcurr = pHead, *pprev = NULL;

    //поки не досягнуто кінця списку
    while (pcurr != NULL) {
        //якщо елемент, який потірбно видалити, перший у списку
        if (!pprev && ((pcurr->m_SBookData.m_nPagesCount < 150) || (pcurr-
>m_SBookData.m_nPublicationYear < 2000))) {
            pHead = pcurr->m_pNext;
            free(pcurr);
            pcurr = pHead;
            continue;
        }
        if ((pcurr->m_SBookData.m_nPagesCount < 150) || (pcurr-
>m_SBookData.m_nPublicationYear < 2000)) {
            pprev->m_pNext = pcurr->m_pNext;
            free(pcurr);
            pcurr = pprev->m_pNext;
            continue;
        }
        pprev = pcurr;
        pcurr = pcurr->m_pNext;
    }

    return pHead;
}

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

// Функція очищення динамічної пам'яті, що була виділена під список книг

// Приймає вказівник на перший елемент списку та ен повертає нічого.

void FreeList(SBookList* pHead) {
    if (!pHead) return;
```

```
    SBookList* pfreeMemory = NULL;

    //поки не досягнуто кінця списку
    while (pHead->m_pNext != NULL) {
        pfreeMemory = pHead;
        free(pfreeMemory->m_pNext);
        pfreeMemory->m_pNext = NULL;
        pHead = pfreeMemory;
    }

    free(pHead);
    pHead = NULL;
}

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

// Функція друкування списку книг у консоль

// Функція приймає вказівник на перший елемент списку та не повертає нічого.
// Друкує список книг у формі таблиці.
void PrintList(SBookList* pHead) {
    if (!pHead) return;

    SBookList* pp = (SBookList*) malloc(sizeof(SBookList));
    PrintLines(COUNT_OF_LINES);
    printf("\n|   Author of the book  |         Book title           | Year of publication |
Number of pages | Price |\n");
    PrintLines(COUNT_OF_LINES);
    while (pHead != NULL) {
        printf("\n|%-23s", pHead->m_SBookData.m_csAuthor);

        printf("|%-30s", pHead->m_SBookData.m_csName);

        printf("|%-21d", pHead->m_SBookData.m_nPublicationYear);

        printf("|%-17d", pHead->m_SBookData.m_nPagesCount);

        printf("|%-7.2lf|\n", pHead->m_SBookData.m_dCost);

        pp = pHead->m_pNext;
        pHead = pp;
    }
    free(pp);
    pp = NULL;

    PrintLines(COUNT_OF_LINES);
}

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

// Функція друкування штрих-пунктирної лінії
```

```
//Приймає кількість надрукованих символів '-' та не повертає нічого
void PrintLines(int count) {
    for (int j = 0; j < count; ++j) printf("- ");
}

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

// Функція видалення книг, вартість яких нижча за середнюв списку
/*Функція приймає вказівник на перший елемент списку та вказівник на вказівник
* на останній елемент списку. Пам'ять видалених елементів очищується
* автоматично
 */
SBookList* DeleteBooksByAvCost(SBookList* pHead, SBookList** pLast) {
    if (!pHead) exit(0);
    SBookList* pcurr = pHead, * pprev = NULL;

    int nCountBooks = 0; double dSumValue = 0.0;
    while (pcurr) {
        dSumValue += pcurr->m_SBookData.m_dCost;
        nCountBooks++;
        pcurr = pcurr->m_pNext;
    }
    double dAvrCost = dSumValue / nCountBooks;
    printf("\n av = %f\n", dAvrCost);
    pcurr = pHead;
    //поки не досягнуто кінця списку
    while (pcurr != NULL) {
        //якщо елемент, який потірбно видалити, перший у списку
        if (!pprev && (pcurr->m_SBookData.m_dCost < dAvrCost)) {
            pHead = pcurr->m_pNext;
            free(pcurr);
            pcurr = pHead;
            continue;
        }
        if (pcurr->m_SBookData.m_dCost < dAvrCost) {
            pprev->m_pNext = pcurr->m_pNext;
            free(pcurr);
            pcurr = pprev->m_pNext;
            continue;
        }
        pprev = pcurr;
        pcurr = pcurr->m_pNext;
    }
    (*pLast) = pprev;

    return pHead;
}

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

Висновок

Під час виконання цієї лабораторної роботи я закріпила знання про документування основні результати проектування та кодування найпростіших програм. Сформувала пакет документів до розробленої раніше власної програми (схематичне зображення використаних структур даних (масив, список), блок-схеми алгоритмів (головної програми, функцій видалення та сортування елементів), код програми, який привела до модульної структури).