**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **КНІТ**
Кафедра **ПЗ**

# ЗВІТ

До лабораторної роботи № 2
**На тему**: "*Документування етапів проектування та кодування програми*"
**З дисципліни**: "Вступ до інженерії програмного забезпечення"

**Лектор**:
доцент кафедри ПЗ
Левус Є.В.

**Виконав**:
студент групи ПЗ-16
Коваленко Д.М.

**Прийняв**:
асистент кафедри ПЗ
Самбір А.А.

«_____» _____ 2022 р.
$\sum$ = _____.....

**Львів — 2022**

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

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

# Теоретичні відомості

**12.** *Які є відомі способи представлення алгоритмів?*

(а) Словесний - подання алгоритму у вигляді певних послідовних пунктів, кожен з яких має певну команду;

(б) Формула - математичний спосіб запису алгоритму;

(в) Розрахункова таблиця - якщо використовується серія розрахунків за однаковими формулами;

(г) Блок-схема - зображення алгоритму геометричними фігурами, що мають певне значення.

**18.** *Які вимоги до запису коментарів у тексті програми?*
Коментарі повинні бути чіткими, читабельними, інформативними. Потрібно дотримуватися орфографічних та граматичних правил під час їх написання.

**28.** *Як записуються класи та їх складові у мові C++?*
Оголошується ключовим словом *class*, яка містить дані (поля) і функції (методи) як свої члени, доступ до якої регулюється трьома специфікаторами доступу: *private*, *public*, *protected* (за умовчанням - *private*). *Private*-члени не доступні за межами класу, вони доступні тільки через методи класу. *Public*-члени є доступними і поза класом.

# Постановка завдання

**Частина I.** 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.

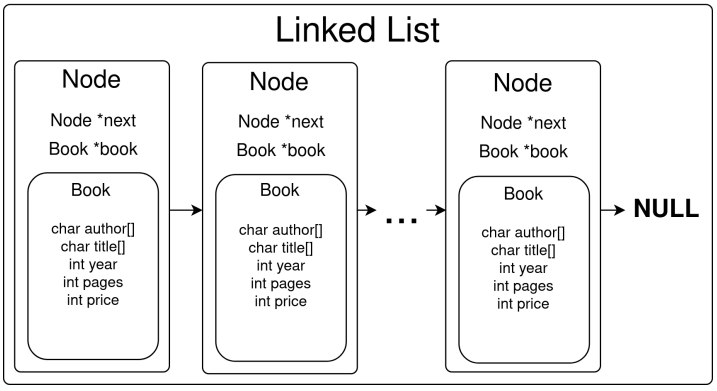
**Частина II.** Сформувати пакет документів до розробленої раніше власної програми:

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.

# Отриманий результат

## Структури даних



## Блок схеми

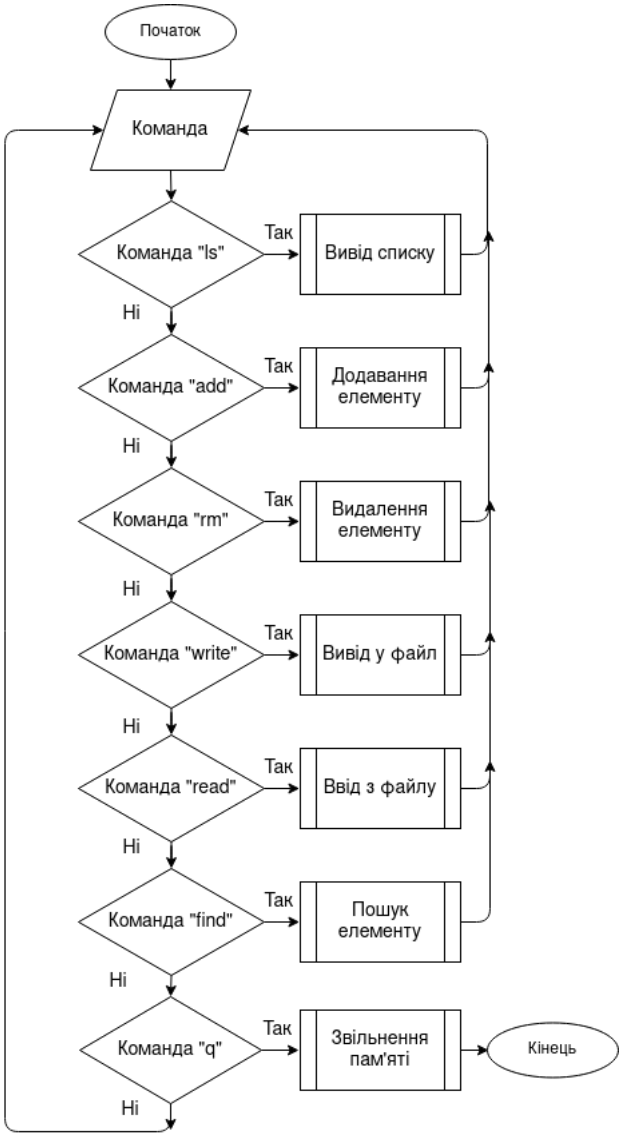



Рис. 1: Блок-схема основної функції

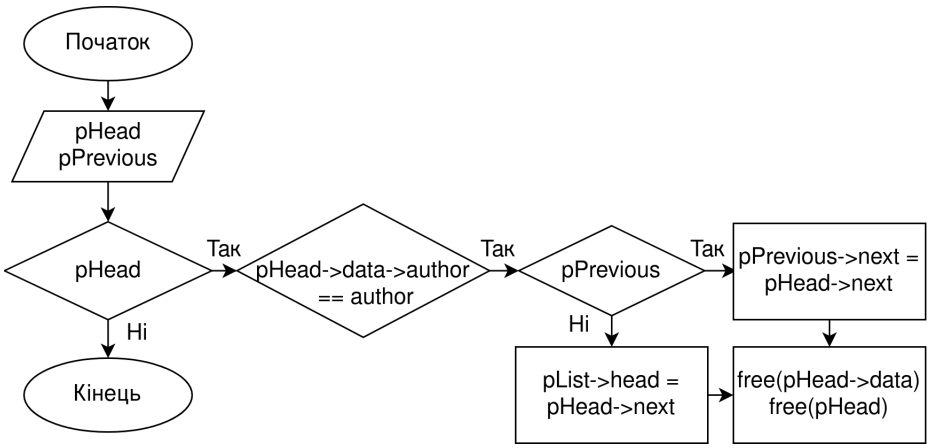



Рис. 2: Блок-схема алгоритму видалення елементу

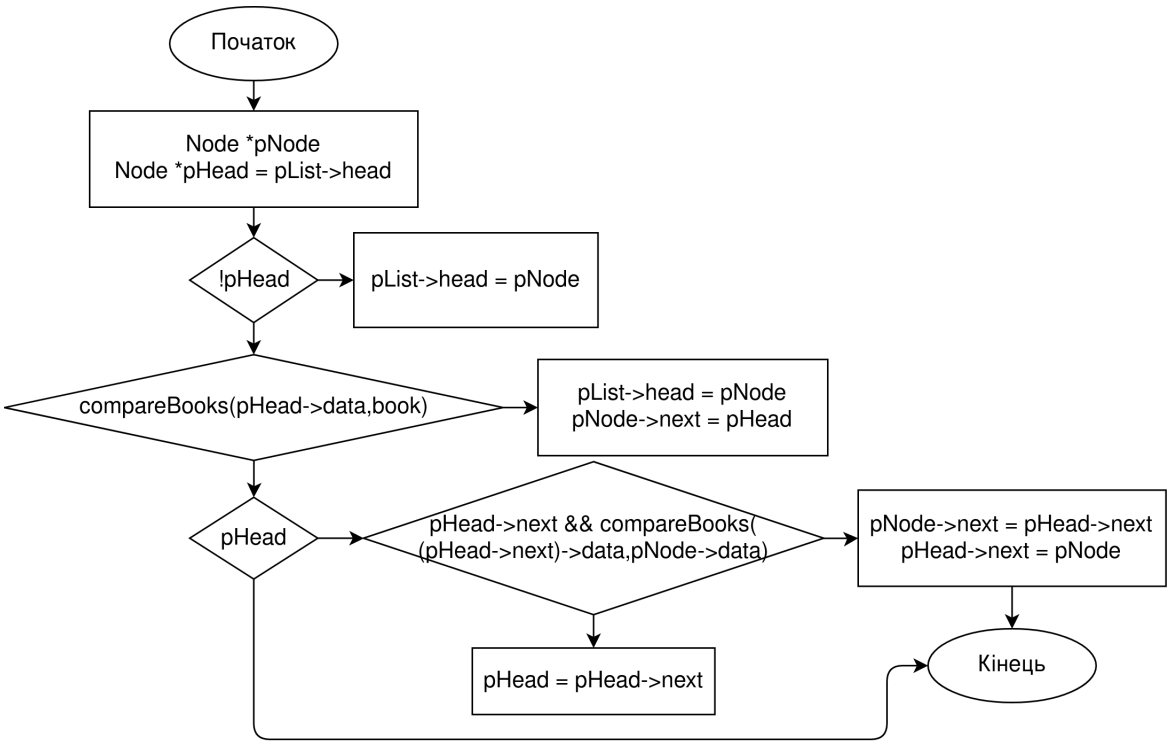



Рис. 3: Блок-схема алгоритму додавання елементу

## Текст програми

Назва файлу: *main.c*

```c
#include <stdio.h>
#include <string.h>
#include "linked_list.h"
#include "book.h"

void readFromFile(LinkedList *pList, char *path) {
    /*
    params:
    LinkedList *pList - linked list pointer
    char *path        - path to file

    Reads items from file and saves it to the linked list
    */

    FILE *pFile = fopen(path,"r");
    char line[256];
    if (pFile != NULL) {
        while (fgets(line, sizeof line, pFile)) {
            Book *book = parseBook(line);
            sortedInsert(pList,book);
        }
        fclose(pFile);
    }
}

void writeToFile(LinkedList *pList, char *path) {
    /*
    params:
    LinkedList *pList - linked list pointer
    char *path        - path to file
```

```
    Writes a linked list items to file
    */

    FILE *pFile = fopen(path, "a");
    Node *pNode = pList->head;

    while (pNode != NULL) {
        char data[1024];
        snprintf(data, 1024, "%s,%s,%d,%d,%d,\n", pNode->data->author,
        pNode->data->title,
        pNode->data->year,
        pNode->data->pages,
        pNode->data->price);
        fputs(data, pFile);
        pNode = pNode->next;
    }
}

int main() {
    /*
    Interactive program management with commands:
    q      - exit program
    ls     - print list of books
    add    - add book to list
    rm     - remove book from list
    write  - write list of books to file
    read   - read list of books from file
    find   - find book in list
    */
    LinkedList list = {NULL};
    char command[64];
    while (1) {
        printf(">>> ");
        fgets(command, sizeof(command), stdin);
        if (strcmp(command,"q\n") == 0) break;
        else if (strcmp(command,"ls\n") == 0) printList(&list);
        else if (strcmp(command,"add\n") == 0) {
            printf(": ");
            fgets(command,sizeof(command),stdin);
            Book *book = parseBook(command);
            sortedInsert(&list,book);
        }
        else if (strcmp(command,"rm\n") == 0) {
            printf(": ");
            fgets(command,sizeof(command), stdin);
            command[strcspn(command,"\n")] = 0;
            rm(&list, command[0]);
        }
        else if (strcmp(command,"write\n") == 0) {
            printf(": ");
            fgets(command,sizeof(command), stdin);
            command[strcspn(command,"\n")] = 0;
            writeToFile(&list, &command[0]);
        }
        else if (strcmp(command,"read\n") == 0) {
            printf(": ");
            fgets(command,sizeof(command), stdin);
            command[strcspn(command,"\n")] = 0;
            freeList(&list);
            readFromFile(&list, &command[0]);
        }
```

```
        else if (strcmp(command,"find\n") == 0) {
            printf(": ");
            fgets(command,sizeof(command), stdin);
            command[strcspn(command,"\n")] = 0;
            find(&list , command[0]);
        }
    }
    freeList(&list);
    return 0;
}
```

Назва файлу: *book.c*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Book {
    char author[64];
    char title[64];
    int year;
    int pages;
    int price;
} Book;

void printBook(Book *book) {
    /*
    params:
    Book *book - Book pointer

    Prints book
    */

    printf("%-20s\t%-25s\t%i\t%i\t%i\n",book->author,book->title,\
    book->year,book->pages,book->price);
}

Book* parseBook(char *line) {
    /*
    params:
    char *line - line that represents a book
    returns:
    Book parsed from line

    Parses Book from line
    */

    Book *book = (Book *)malloc(sizeof(Book));
    if (!book) return NULL;
    int j = 0;
    for (char *field = strtok(line,","); field; field = strtok(NULL,",")) {
        ++j;
        switch (j % 10) {
            case 1: stpcpy(book->author, field);
            case 2: stpcpy(book->title, field);
            case 3: book->year = atoi(field);
            case 4: book->pages = atoi(field);
            case 5: book->price = atoi(field);
        }
    }
    return book;
}
```

```
char getAuthor(Book *book) {
    /*
    params:
    Book *book - Book pointer
    returns:
    Book's author first char
    */

    return book->author[0];
}

int compareBooks(Book *lValue,Book *rValue) {
    /*
    params:
    Book *lValue - Book pointer
    Book *rValue - Book pointer
    returns:
    Comparision result

    Compares books by the first char of author name
    */

    return lValue->author[0] >= rValue->author[0]?1:0;
}

void printTitle() {
    /*
    Prints title for table
    */

    printf("%-20s\t%25s\t%s\t%s\t%s\n","Author","Title","Year","Pages","Price");
}
```

Назва файлу: *linked_list.c*

```
#include <stdlib.h>
#include <string.h>
#include "book.h"

typedef struct Node {
    Book *data;
    struct Node *next;
} Node;

typedef struct LinkedList {
    Node *head;
} LinkedList;

LinkedList* sortedInsert(LinkedList *pList, Book *book) {
    /*
    params:
    LinkedList *pList - linked list pointer
    Book *book        - Book pointer

    Inserts a book into the linked list so that the list remains sorted
    */

    Node *pNode = (Node*)malloc(sizeof(Node));
    if (!pNode) return NULL;
    pNode->data = book;
    pNode->next = NULL;
```

```
    Node *pHead = pList->head;
    if(!pHead) {
        pList->head = pNode;
        return pList;
    }
    if (compareBooks(pHead->data,book)) {
        pList->head = pNode;
        pNode->next = pHead;
        return pList;
    }
    while(pHead){
        if(
        (
        compareBooks(pNode->data,pHead->data)
        && !(pHead->next)
        )
        ||
        (
        compareBooks(pNode->data,pHead->data)
        &&
        (
        pHead->next
        && compareBooks((pHead->next)->data,pNode->data)
        )
        )
        ) {
            pNode->next = pHead->next;
            pHead->next = pNode;
            break;
        } else pHead = pHead->next;
    }
    return pList;
}

void rm(LinkedList *pList, char author) {
    /*
    params:
    LinkedList *pList - linked list pointer
    char author       - first char of author

    Recursively remove items from list
    */

    Node *pHead = pList->head;
    Node *pPrevious = NULL;
    while (pHead) {
        if(getAuthor(pHead->data) == author) {
            if(pPrevious) pPrevious->next = pHead->next;
            else pList->head = pHead->next;
            free(pHead->data);
            free(pHead);
            rm(pList,author);
            break;
        }
        pPrevious = pHead;
        pHead = pHead->next;
    }
}

Book* find(LinkedList *pList, char author) {
    /*
```

```
    params:
    LinkedList *pList - linked list pointer
    char author       - first char of author

    Prints all found books
    */

    Node *pHead = pList->head;
    while (pHead) {
        if (getAuthor(pHead->data) == author) printBook(pHead->data);
        pHead = pHead->next;
    }
}

void printList(LinkedList *pList) {
    /*
    params:
    LinkedList *pList - linked list pointer

    Prints all items of list
    */

    printTitle();
    Node *pNode = pList->head;
    while (pNode) {
        printBook(pNode->data);
        pNode = pNode->next;
    }
}

void freeList(LinkedList *pList) {
    /*
    params:
    LinkedList *pList - linked list pointer

    Frees all dynamicaly allocated memory
    */

    Node *pNode = pList->head;
    Node *pTemp;
    while (pNode){
        free(pNode->data);
        pTemp = pNode->next;
        free(pNode);
        pNode = pTemp;
    }
}
```

Назва файлу: *book.h*

```
#ifndef BOOK_H
#define BOOK_H

typedef struct Book {
    char author[64];
    char title[64];
    int year;
    int pages;
    int price;
} Book;

void printBook(Book *book);
```

```
Book* parseBook(char *line);
char* getAuthor(Book *book);
int compareBooks(Book *lValue, Book *rValue);
void printTitle();

#endif
```

Назва файлу: *linked_list.h*

```
#ifndef LINKED_LIST_H
#define LINKED_LIST_H
#include "book.h"

typedef struct Node {
    Book *data;
    struct Node *next;
} Node;

typedef struct LinkedList {
    Node *head;
} LinkedList;

LinkedList* sortedInsert(LinkedList *list, Book *book);
LinkedList* rm(LinkedList *pList, char author);
Book* find(LinkedList *pList, char author);
Book* edit(LinkedList *pList, char author);
void printList(LinkedList *list);
void freeList(LinkedList *list);

#endif
```

## Висновок

На лабораторній роботі я навчився документувати основні результати етапів проектування та кодування найпростіших програм.