НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЗВІТ

про виконання лабораторної роботи №2

Виконав: Климок Н.І.
студент групи ПЗ - 16

_____
(дата виконання)

Львів - 2022

Тема. Документування етапів проектування та розробки.

Мета. Навчитися документувати основні результати етапу проектування та розробки найпростіших програм.

ТЕОРЕТИЧНІ ВІДОМОСТІ

8. Переваги/недоліки масивів і однозв'язних списків

Масив

+ *Швидкий доступ (константний час)*
+ *Легкість використання*
- *Обмежений розмір, необхідне копіювання для зміни розміру*

Однозв'язний список

+ *Динамічний розмір*
- *Повільний (пошук за O(n))*

27. Яка послідовність методів у кожній секції класу у мові C++? Навести приклад.

Згідно з C++ Style Guide компанії Google, послідовність секцій повинна бути такою:

1. *Секція public*
2. *Секція protected*
3. *Секція private*

Пусті секції не повинні бути прописані.

У кожній секції повинен бути такий порядок визначень:

1. *Типи і аліаси (typedef, using, enum, влаштовані структури і класи)*
2. *Статисні константи*
3. *Factory функції*
4. *Constructor-и і оператори присвоєння*
5. *Деструктори*
6. *Усі інші функції (static, non-static функції-члени і дружні функції)*
7. *Поля класу (static і non-static)*

23. Які правила запису назв функцій? Навести п'ять прикладів.

1. *Гетери і сетери повинні відповідати назві поля і мати відповідний префікс (count, set_count, get_count)*
2. *Функції повинні називатися snake-case'ом.*
3. *Функції повинні називатися з маленької літери*
4. *Аргументи функції повинні починатися з малої літери і називатися snake-case'ом.*
5. *Ім'я функції чи метода має починатися на дієслово.*

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:

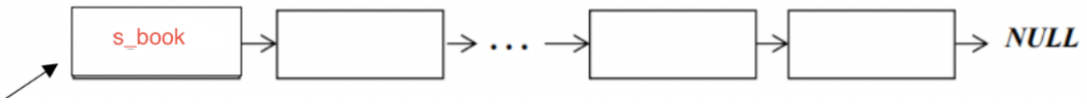
1. схематичне зображення структур даних, які використовуються для збереження інформації ;
2. блок-схема алгоритмів – основної функції й двох окремих функцій-підпрограм (наприклад, сортування та редагування);
3. текст програми з коментарями та оформлений згідно вище наведених рекомендацій щодо забезпечення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

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

Виконання завдання

1. Схематичне зображення структури даних:

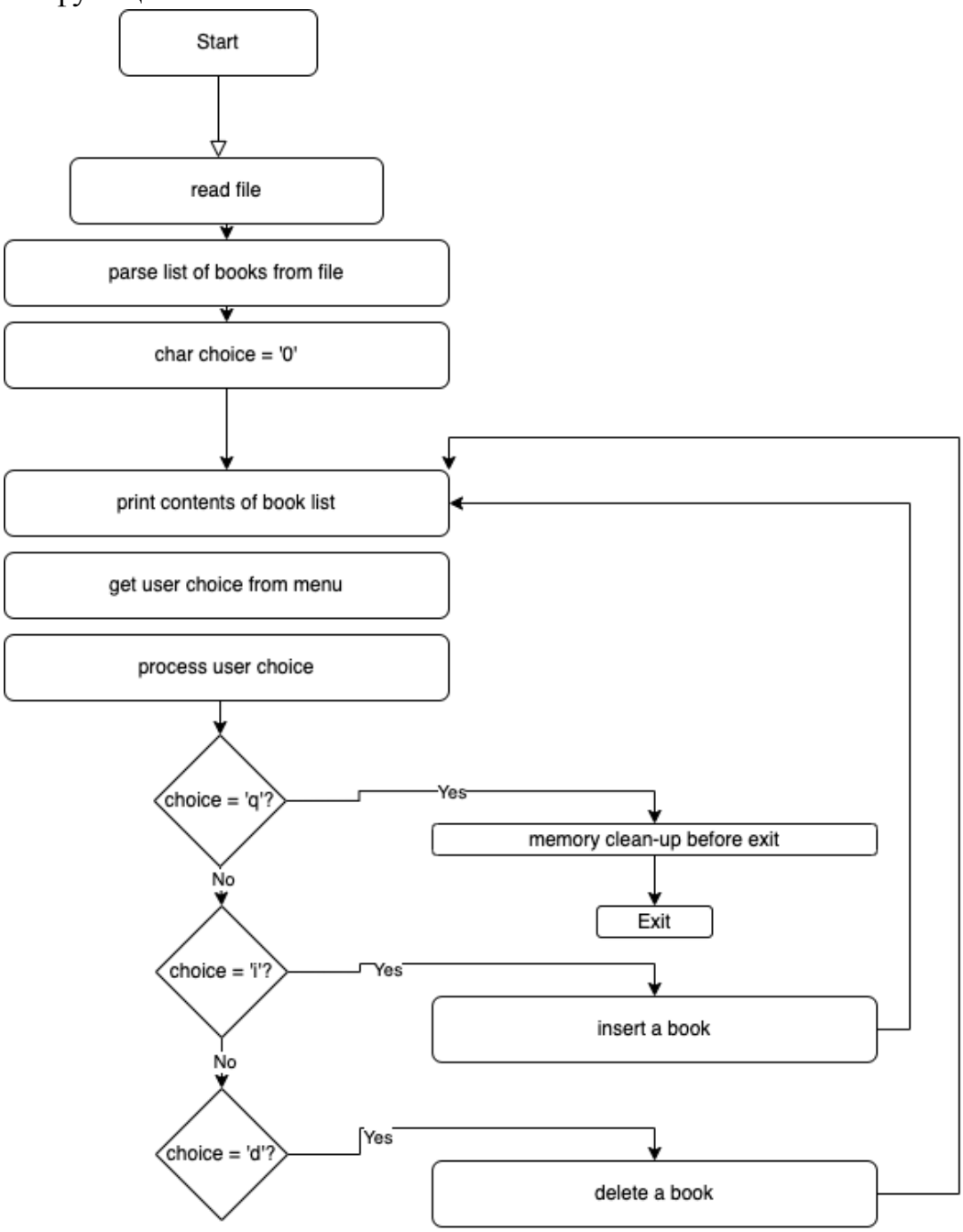
Зв'язний список – одна з найважливіших структур даних, в якій елементи лінійно впорядковані, але порядок визначається не номерами елементів, а вказівниками, які входять в склад елементів списку та вказують на наступний за елемент (в однозв'язних списках (ланцюг), рис. 4) або на наступний та попередній елементи (в двозв'язних списках, рис. 5).



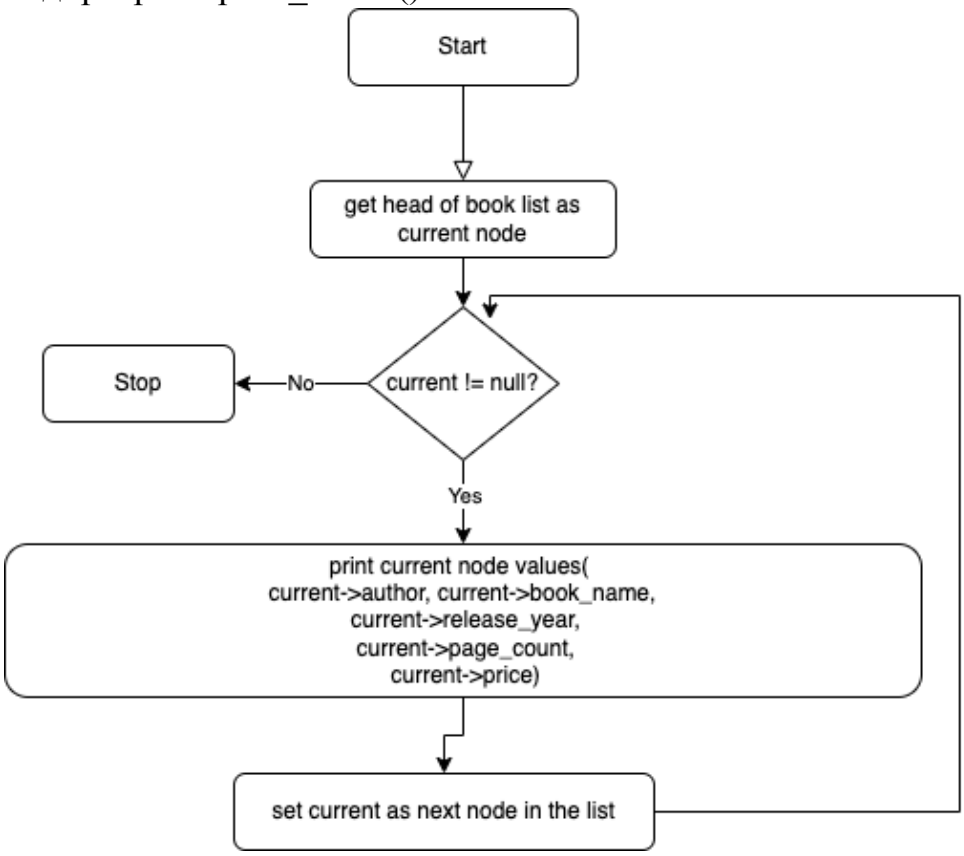

```
typedef struct s_book {
      char *author;
      char *book_name;
      int release_year;
      int page_count;
      double price;
      struct s_book *next;
} t_book;
```

2. Блок-схема алгоритмів

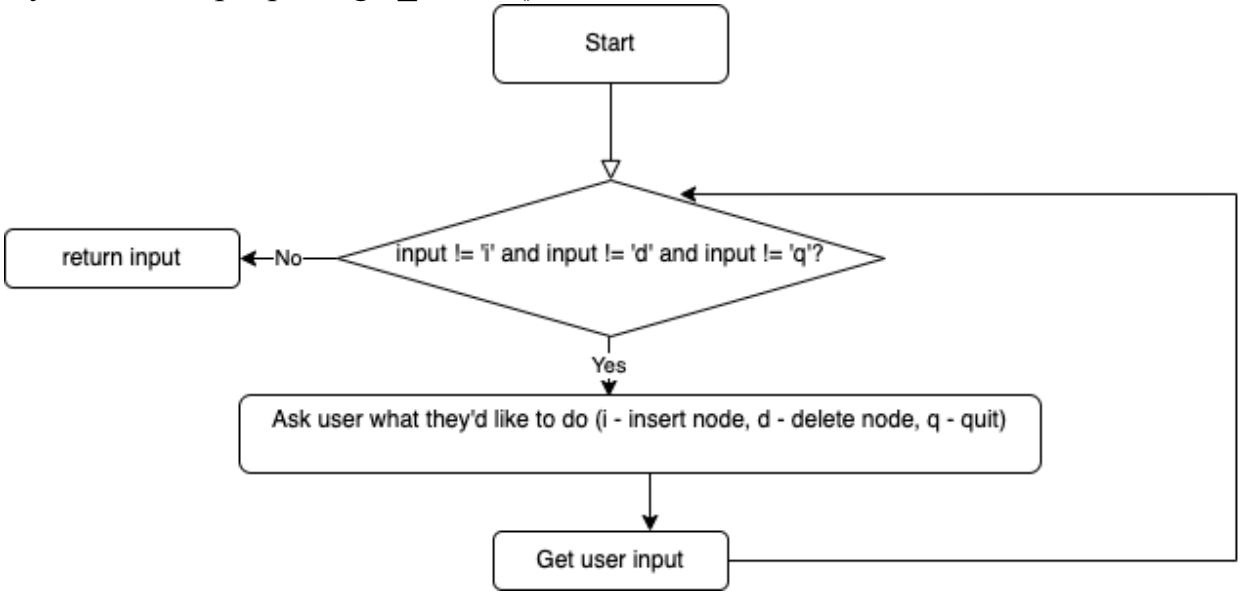
Основна функція:



Функція-підпрограма print_books():



Функція-підпрограма get_choice():



3. Текст програми з коментарями:

header.h:

```
#ifndef MY_HEADER

#define MY_HEADER

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

typedef struct s_book {
     char *author;
     char *book_name;
     int release_year;
     int page_count;
     double price;
     struct s_book *next;
} t_book;

// Input
char *read_file(char *filename);
void insert_book(t_book **head_ref);

// Output
void print_books(t_book *head);

// Parsing
t_book *parse_books(char *input);
void get_book_from_str(char *str, t_book *curr_student);

//List
void add_book(t_book **head_ref, t_book *new);
void delete_books(t_book **head_ref);

#endif
```

list.c:

```
#include "header.h"

// util to compare contents
static int strcicmp(char const *a, char const *b)
{
     for (;; a++, b++) {
          int d = tolower((unsigned char)*a) - tolower((unsigned
          char )*b);
          if (d != 0 || !*a)
               return d;
     }
}

// deletes from list and frees memory of matching objects
static void remove_where_name_starts_on_d(t_book** head_ref)
{
     t_book *temp = *head_ref, *prev;
     while (temp && temp->book_name[0] == 'D' ||
     temp->book_name[0] == 'd')
     {
          *head_ref = temp->next;
          free(temp->book_name);
          free(temp->author);
          free(temp);
          temp = *head_ref;
     }
     while (temp)
     {
          while (temp && temp->book_name[0] != 'D' &&
     temp->book_name[0] != 'd')
          {
               prev = temp;
               temp = temp->next;
          }
          if (!temp)
               Return;
          prev->next = temp->next;
          free(temp->book_name);
          free(temp->author);
          free(temp);
          temp = prev->next;
     }
}
```

```
// adds book in a sorted fashion
void add_book(t_book** head_ref, t_book* new_node)
{
     t_book *current;
     if (*head_ref == NULL || strcicmp((*head_ref)->book_name,
new_node->book_name) >= 0) {
          new_node->next = *head_ref;
          *head_ref = new_node;
     }
     else {
          current = *head_ref;
          while (current->next != NULL &&
     strcicmp(current->next->book_name,
          new_node->book_name) < 0) {
               current = current->next;
          }
          new_node->next = current->next;
          current->next = new_node;
     }
}

void delete_books(t_book **head_ref)
{
     remove_where_name_starts_on_d(head_ref);
}
```

input.c:

```
#include "header.h"

char *read_file(char *filename)
{
     FILE *fp = fopen(filename, "r");
     char buf[10000];
     int count = 0;
     while ((buf[count] = getc(fp)) != EOF)
     {
          Count++;
     }
     fclose(fp);
     return strndup(buf, count);
}
```

```
void insert_book(t_book **head_ref)
{
     char input[100] = {'\0'};
     t_book *new_book = (t_book *) malloc(sizeof(t_book));
     printf("Please, enter book data (Format: Author name, book
name, release year, page count, price):\n");
     fgets(input, 100, stdin);
     get_book_from_str(input, new_book);
     add_book(head_ref, new_book);
}
```

output.c:

```
#include "header.h"

void print_books(t_book *head)
{
     t_book *current = head;
     while (current)
     {
          printf("%s | %s | %d | %d | %lf\n", current->author,
          current->book_name, current->release_year,
          current->page_count, current->price);
          current = current->next;
     }
}
```

parsing.c:

```
#include "header.h"

void get_book_from_str(char *str, t_book *curr_book) {
     char *endptr;
     char *linedup = strdup(str);
     char *a_name = strdup(strtok(linedup, ","));
     char *b_name = strdup(strtok(NULL, ","));
     int r_year = atoi(strtok(NULL, ","));
     int p_count = atoi(strtok(NULL, ","));
     double pr = strtod(strtok(NULL, ","), &endptr);

     curr_book->author = a_name;
     curr_book->book_name = b_name;
     curr_book->release_year = r_year;
     curr_book->page_count = p_count;
     curr_book->price = pr;
     curr_book->next = NULL;
     free(linedup);
```

```
}

t_book *parse_books(char *input)
{
     t_book *head = (t_book *) malloc(sizeof(t_book));
     t_book *prev = head;
     t_book *next;
     char *save_ptr;
     char *curr_line = strtok_r(input, "\n", &save_ptr);

     get_book_from_str(curr_line, head);
     curr_line = strtok_r(NULL, "\n", &save_ptr);
     while (curr_line)
     {
          next = (t_book *) malloc(sizeof(t_book));
          get_book_from_str(curr_line, next);
          prev->next = next;
          prev = next;
          curr_line = strtok_r(NULL, "\n", &save_ptr);
     }
     return head;
}
```

main.c:

```
#include "header.h"

void cleanup(t_book *head, char *input)
{
     t_book *next;
     if (input)
     free(input);
     while (head)
     {
          next = head->next;
          free(head->book_name);
          free(head->author);
          free(head);
          head = next;
     }
}
```

```
char get_choice()
{
     char c = '0';
     while (c != 'i' && c != 'd' && c != 'q')
     {
          printf("What would you like to do? (i - insert node, d
          - delete
          node, q - quit): ");
          c = getc(stdin);
          getc(stdin);
     }
     return c;
}

void process_choice(char c, t_book **head_ref, char *input)
{
     switch (c) {
          case 'q':
               cleanup(*head_ref, input);
               exit(0);
               break;
          case 'i':
               insert_book(head_ref);
               break;
          case 'd':
               delete_books(head_ref);
               break;
     }
}

int main(void)
{
     char *input = read_file("input.txt");
     t_book *head = parse_books(input);
     char choice = '0';
     while (1)
     {
          print_books(head);
          choice = get_choice();
          process_choice(choice, &head, input);
     }
     cleanup(head, input);
}
```

Висновок

У цій лабораторній роботі я навчився документувати основні результати етапу проектування та розробки найпростіших програм.