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

з дисципліни «ВСТУП ДО ІНЖЕНЕРІЇ ПЗ»

Лектор:
доцент кафедри ПЗ
Левус Є. В.

Виконав:
студ. групи ПЗ-16
Фур Я.М.

Прийняв:
Самбір А. А.
«___» _____ 2020 р.
∑ = ____ _____

Львів – 2020

Тема роботи: Документування етапів проектування та кодування програми.

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

Теоретичні відомості

11) Властивості алгоритмів.

Алгоритми мають такі властивості:

- Зрозумілість (Виконавець повинен розуміти кожну команду алгоритму).
- Детермінованість (однозначність) (Все має бути однозначно, ніякої двозначності в командах).
- Скінченність (результативність) (Результата має досягатися за скінченну кількість команд).
- Дискретність (Поділ на послідовність дій).
- Масовість (універсальність) (Алгоритм можна використовувати для різних задач).

15) Скільки входів і виходів має блок розгалуження?

Блок розгалуження має один вхід і два виходи. Через вхід передаються вхідні дані, і залежно від них і своєї умови блок вибирає один з двох виходів.

29) Як можна підвищити продуктивність роботи на етапі кодування?

Писати код за єдиним стандартом, зручним для читання. Код має бути якнайпростішим, не ускладненим зайвими діями і командами. Чим простіше код, тим менше в ньому помилок.

Постановка завдання

1) Привести раніше розроблену програму з дисципліни “Основи програмування” до модульної структури.

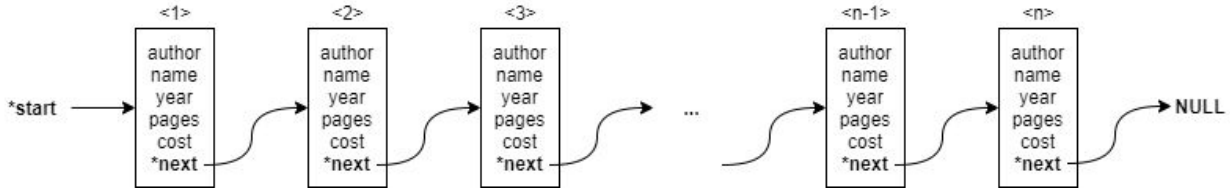
2) Сформувати пакет документів до програми:

- Схематичне зображення структур даних, які використовуються для збереження інформації.
- Блок-схема алгоритмів (до головної функції та 2 інших функцій)
- Текст програми з коментарями та оформлений згідно наведених правил.

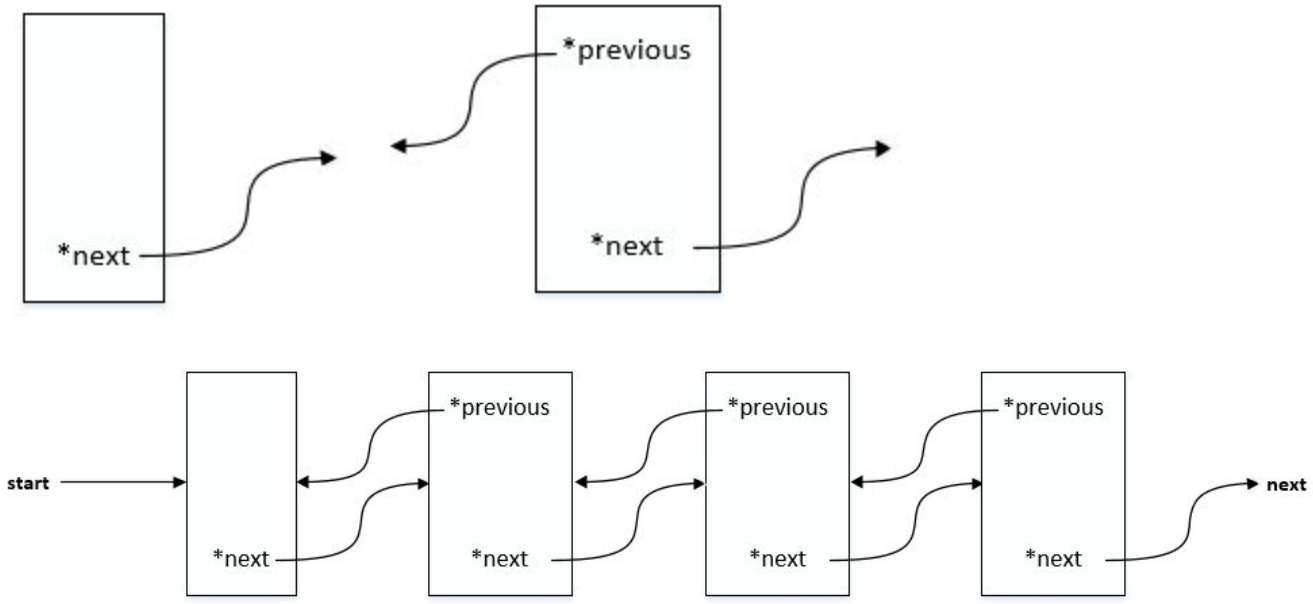
3) У редакторі MS Visio розробити зразки фігур, які були використані для схематичного зображення структур, як готові трафарети для використання.

Отримані результати

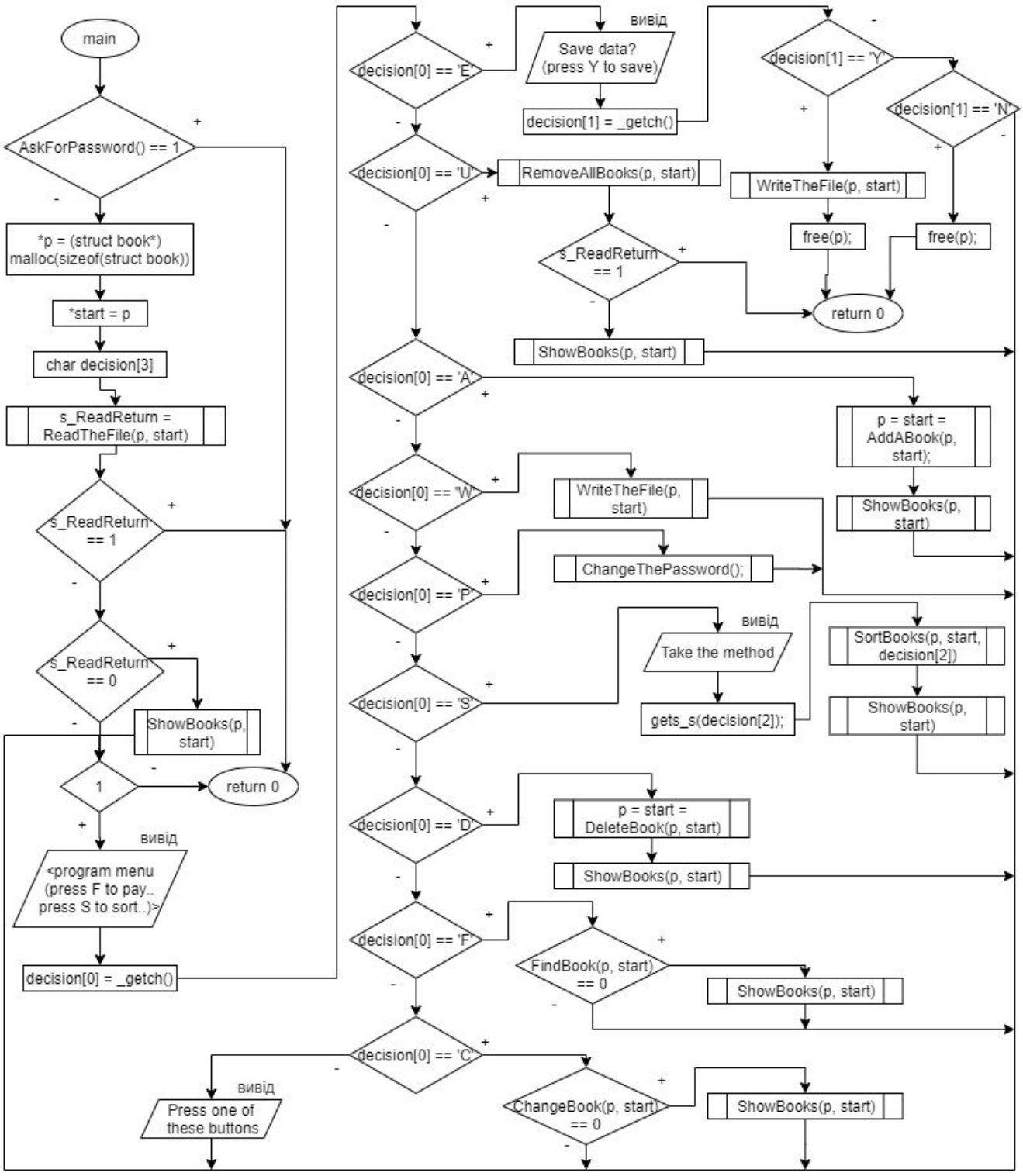
Схематичне зображення використаної структури даних:



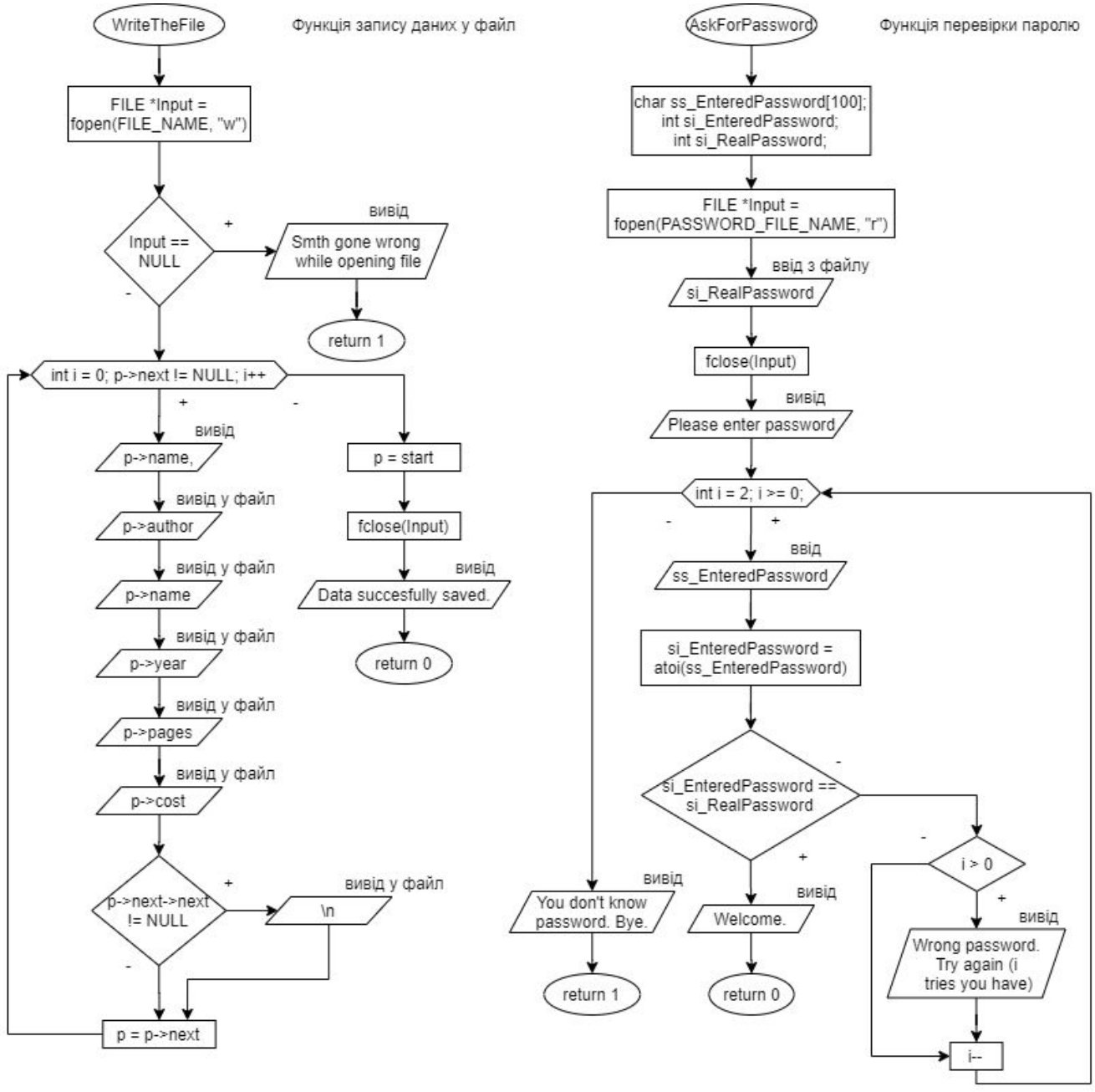
Створені зразки фігур, які використовуються в структурі і які можна використовувати надалі:



Блок-схема головної функції:



Блок-схеми 2 функцій-підпрограм:



Код програми:

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

```
#define FILE_NAME "C:/Users/User/source/repos/VIPZ_Lab_2.0/ForVIPZ_Lab_2.0.txt"  // основний файл для збереження/читання
#define PASSWORD_FILE_NAME "C:/Users/User/source/repos/VIPZ_Lab_2.0/TOP SECRET.txt"  // файл з паролем
#define swapint(x,y,tmp) tmp = x; x = y, y = tmp;  // свапає цілі числа
#define swapstr(x,y,tmp) strcpy(tmp, x); strcpy(x, y); strcpy(y, tmp);  // свапає стрічки

int ReadTheFile(struct book*, struct book*);  // зчитує книги з файлу і створює нову структуру
int WriteTheFile(struct book*, struct book*);  // записує книги в файл
int AskForPassword();  // питає користувача пароль
void ShowBooks(struct book*, struct book*); // виводить дані на екран
struct book *AddABook(struct book*, struct book*);  // додає книгу в бібліотеку
void RemoveAllBooks(struct book*, struct book*);  // видаляє всі книги для перезапису
void ChangeThePassword(); // змінити пароль
void SortBooks(struct book*, struct book*, int);  // сортування книг
struct book *DeleteBook(struct book*, struct book*); // видалення книг
int FindBook(struct book*, struct book*);  // знаходить книгу
int ChangeBook(struct book*, struct book*); // змінює книгу

char g_tempStr[200];  // змінна для збереження стрічки, яка потрібна при зчитуванні. Також використовується як тимчасова в інших функціях
int g_NumBooks = 0;  // буде зберігати значення кількості книжок впродовж усієї програми
struct book  // сама структура
{
  char author[100];
  char name[100];
  int year;
  int pages;
  int cost;
  struct book *next;  // однозв*язний список
};

int main()
{
  if (AskForPassword() == 1) {  // звісно програма захищена
    return 0;
  }
   struct book *p = (struct book*)malloc(sizeof(struct book));
  struct book *start = p; // створюємо структури
  char decision[3];  // в десижн буде записуватися вибір користувача що робити з книгами

  int s_ReadReturn = ReadTheFile(p, start);
  if (s_ReadReturn == 1 || s_ReadReturn == 2) return 0;
      // перевірка чи файл добре прочитаний
  else if (s_ReadReturn == 0) ShowBooks(p, start);
  while (1) {  // це великий цикл який буде повторюватися поки користувач не вимкне програму
        printf("\n\tWhat do you want to do? You can:\n\tUpdate from file and show data (press U)\n\tSort library (press S);\n\t"
        "Add new book (press A)\n\tDelete book (press D)\n\tFind a book (press F)\n\tChange a book (press C)\n\t"
        "Write data into the file (press W)\n\tPassword change(press P)\n\tExit (press E)\n\n");  // головне меню програми
        decision[0] = _getch();
        if (decision[0] == 'E' || decision[0] == 'e') {
        // робота в меню: вихід
          printf("\nSave data? (press Y to save) (press N to exit without saving) (press smth else to cancel and stay in program)\n");
          decision[1] = _getch();
          if (decision[1] == 'Y' || decision[1] == 'y') {  // зберегти
             WriteTheFile(p, start);
                while (p) {
                  if (p) start = p->next;
                  free(p);
                  if (p) p = start;
                }
```

```
                return 0;
            } else if (decision[1] == 'N' || decision[1]
== 'n') {        // не зберігати
              while (p) {
                if (p) start = p->next;
                     free(p);
                     if (p) p = start;
              }
              return 0;
            }
          } else if (decision[0] == 'U' || decision[0] ==
'u') {   // оновити дані
            RemoveAllBooks(p, start);
            s_ReadReturn = ReadTheFile(p, start);
            if (s_ReadReturn == 1) return 0;
            ShowBooks(p, start);
          } else if (decision[0] == 'A' || decision[0] ==
'a') {   // додати книгу
            p = start = AddABook(p, start);
            ShowBooks(p, start);
          } else if (decision[0] == 'W' || decision[0] ==
'w') {   // записати в файл
            WriteTheFile(p, start);
          } else if (decision[0] == 'P' || decision[0] ==
'p') {   // змінити пароль
            ChangeThePassword();
          } else if (decision[0] == 'S' || decision[0] ==
's') {   // сортувати
             printf("Take the method:\n1 - Author,\n2 -
Name,\n3 - Years,\n4 - Pages,\n5 - Cost,\n");
            gets_s(g_tempStr);
            decision[2] = atoi(g_tempStr);
            SortBooks(p, start, decision[2]);
            ShowBooks(p, start);
          } else if (decision[0] == 'D' || decision[0] ==
'd') {   // видалити книгу
            p = start = DeleteBook(p, start);
            ShowBooks(p, start);
          } else if (decision[0] == 'F' || decision[0] ==
'f') {   // знайти книгу
             if (FindBook(p, start) == 0) ShowBooks(p,
start);
          } else if (decision[0] == 'C' || decision[0] ==
'c') {   // змінити книгу
            if (ChangeBook(p, start) == 0) ShowBooks(p,
start);
          } else printf("Press one of these buttons\n");
          // повтор меню
    }
    return 0;
}

// зчитує книги з файлу і записує їх в структуру, яку створює
int ReadTheFile(struct book *p, struct book *start)
{
    FILE *Input = fopen(FILE_NAME, "r");
    if (Input == NULL) {    //    відкриваємо    і
зразу ж перевіряємо файл
      printf("Smth gone wrong while opening file (file
\"%s\"   does   not   exist   or   program   has   no
allowance)...\n", FILE_NAME);
    return 1;
    } else {
          char *mark;      // марк - кома, по якій орієнтуємося при зчитуванні з файлу (вона розділяє дані)
          for  (;  !feof(Input);  g_NumBooks++)  {    //
поки не досягнемо кінець файлу
            if (fgets(g_tempStr, 199, Input) == NULL) {
               printf("The library has a problem, it can
be empty.\n");
                  return 1;
            }
            mark = strtok(g_tempStr, ",");   //
зчитуємо дані з файла і записуємо в тимчасову змінну по одній книзі до \n
            for (int j = 1; j < 6; j++) {    //
орієнтуючись по комах
                  switch (j)
                  {
                      case  1:  strcpy(p->author,  mark);
break;
                       case  2:  strcpy(p->name,  mark);
break;  // записуємо дані в динамічні структури
                    case 3: p->year = atoi(mark); break;
                      case  4:  p->pages  =  atoi(mark);
break;
                    case 5: p->cost = atoi(mark); break;
                  }
                  mark = strtok(NULL, ",");
            }
            if (g_NumBooks == 0) start = p;  //   якщо це перший запис в перший елемент, то він стає стартовим
                  struct    book    *pp    =    (struct
book*)malloc(sizeof(struct book)); //   наступний елемент
```

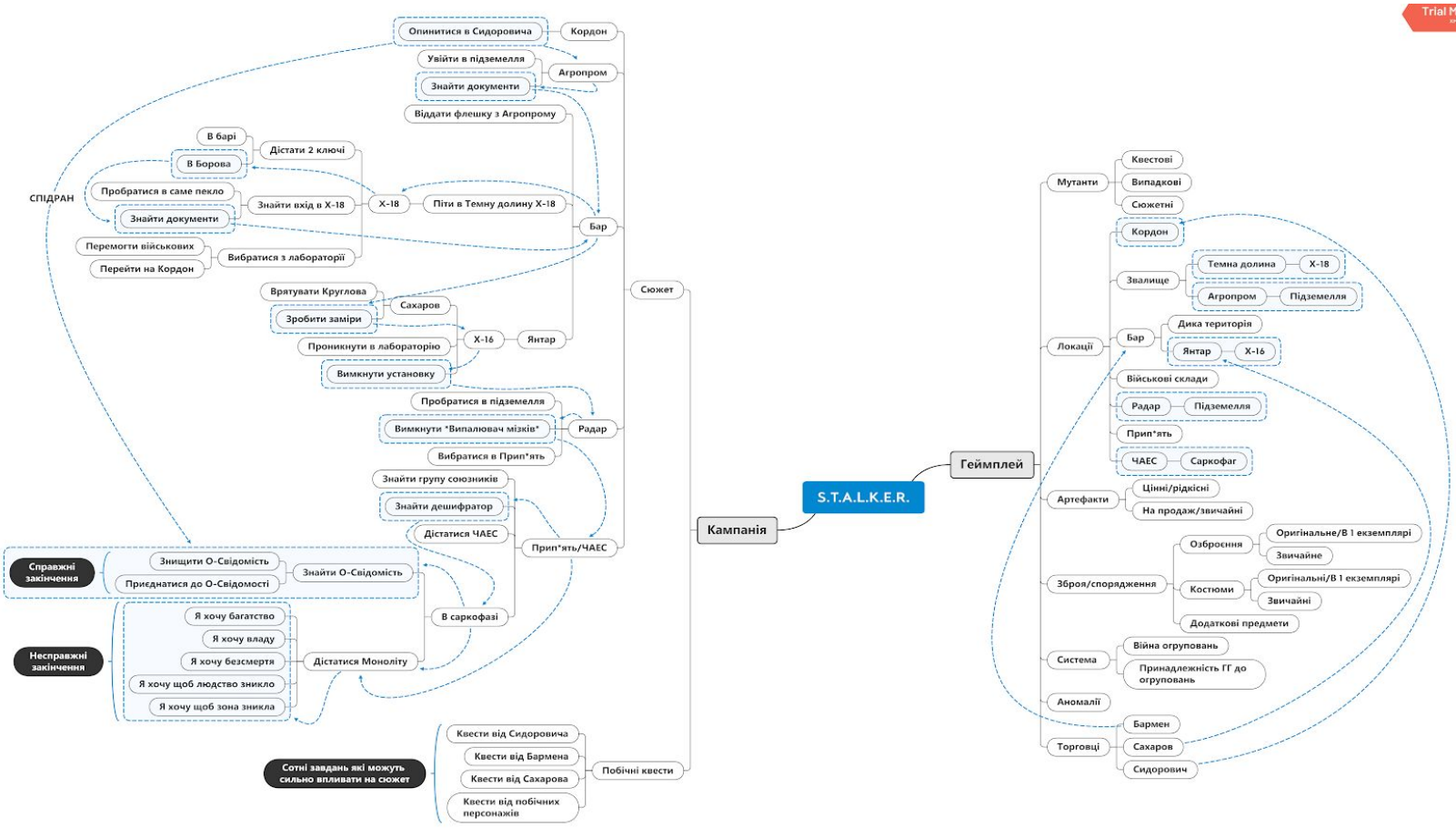
```
          p->next = pp;
          p = pp;        //           створюємо наступні елементи структури
        }
        p->next = NULL;  //          закриваємо останній елемент
        p = start;       //    і    переміщаємо основний вказівник на початок структури
        fclose(Input);
  }
  return 0;
}

// записує книги в файл
int WriteTheFile(struct book *p, struct book *start)
{
  FILE *Input = fopen(FILE_NAME, "w");
  if (Input == NULL) {    //    відкриваємо і зразу ж перевіряємо файл
        printf("Smth gone wrong while opening file (file \"%s\" can't be created or program has no allowance)...\n", FILE_NAME);
        return 1;
  } else {
    for (int i = 0; p->next != NULL; i++) {
          printf("%s,\n", p->name);
          fprintf(Input, "%s,", p->author);        // записуємо всі дані по порядку в файл
          fprintf(Input, "%s, ", p->name);
          fprintf(Input, "%d, ", p->year);
          fprintf(Input, "%d, ", p->pages);
          fprintf(Input, "%d", p->cost);
            if(p->next->next  !=  NULL)  fprintf(Input, "\n");
          p = p->next;   // переміщаємось по елементах
        }
        p = start;       //         в         кінці переміщаємо основний вказівник на початок структури
        fclose(Input);
        printf("\n Data succesfully saved.\n");
  }
  return 0;
}

// вимагає пароль, який бере з спеціального файлу
int AskForPassword()
{
  char ss_EnteredPassword[100];    // використовується      для      вводу паролю
  int si_EnteredPassword; // щоб перетворити стрічку в число
  int si_RealPassword;    // для зчитування паролю з файлу
  FILE *Input = fopen(PASSWORD_FILE_NAME, "r");
  fscanf(Input, "%d", &si_RealPassword);
  fclose(Input);
  printf("Please enter password\t");
  for (int i = 2; i >= 0;) {
        gets_s(ss_EnteredPassword);        //    ввід паролю
        si_EnteredPassword  =  atoi(ss_EnteredPassword);
        // претворення
        if  (si_EnteredPassword  ==  si_RealPassword)  {
        // перевірка
          printf("Welcome.\n");
          return 0;
        } else { // відмова
          if (i > 0) printf("Wrong password. Try again (%d tries you have)\t", i);
          i--;
        }
  }
  printf("You don't know password. Bye.\n"); // якщо 3 спроби вичерпано
  return 1;
}

// виводить список книг на екран
void ShowBooks(struct book *p, struct book *start)
{
  printf("\t%d books found", g_NumBooks);
  printf("\n\tHere's updated library:\n\n");
  printf("# %-35s%-35s%-10s%-10s%-10s\n\n\n", "Author", " Name", "Year", "Pages", "Cost");
  for (int i = 0; p->next; i++) {
        printf("%d", i + 1);
        printf(" %-35s", p->author);
        printf("%-35s", p->name);
        printf("%-10d", p->year); // просто і по порядку
        printf("%-10d", p->pages);
        printf("%-10d", p->cost);
        printf("\n\n");
        p = p->next;
```

```
  }
  p = start;
}

// додавння нової книги
struct book *AddABook(struct book *p, struct book *start)
{
         struct book* newBook = (struct book*)malloc(sizeof(struct book));
   newBook->next = start;    // створюємо ще 1 книгу
  start = newBook;
  p = start;
  printf("\n\nEnter the author of new book\t");
  gets_s(p->author);
  printf("\n\nEnter the name of new book\t");
   strcpy(p->name, " ");   // додаємо до p->name пробіл спочатку, бо кожна назва починаєстья з пробілу
  gets_s(g_tempStr);
  strcat(p->name, g_tempStr);
  printf("\n\nEnter the year of new book\t");
  gets_s(g_tempStr);
  p->year = atoi(g_tempStr);
  printf("\n\nEnter pages of new book\t\t");
  gets_s(g_tempStr);
  p->pages = atoi(g_tempStr);
  printf("\n\nEnter the cost of new book\t");
  gets_s(g_tempStr);
  p->cost = atoi(g_tempStr);
  g_NumBooks++;
  return start;
}

// видаляє список книг з пам*яті для перезапису
void RemoveAllBooks(struct book *p, struct book *start)
{
  p = start->next;
  struct book *pp = p;
  while (p->next != NULL) {
        p = p->next;
        free(pp);        // і одразу звільняє пам*ять
        pp = p;
        g_NumBooks--;
  }
  free(pp);      //      для      останнього елементу
  g_NumBooks--;
  p = start;
}

// змінює пароль у файлі
void ChangeThePassword()
{
  AskForPassword();       //    спершу    питає дійсний пароль
  int si_NewPassword;
  char ss_NewPassword[100];
  FILE *Input = fopen(PASSWORD_FILE_NAME, "w");
  printf("Please enter new password\t");
  gets_s(ss_NewPassword);
  si_NewPassword = atoi(ss_NewPassword);    // перетворює стрічки у цифри
  fprintf(Input, "%d", si_NewPassword);
  fclose(Input);
  printf("\tPassword changed\n");
}
// сортує по чому завгодно
void SortBooks(struct book *p, struct book *start, int method)
{
   int more; // для запису сортування в процесі
  for (int i = 0, tempint; i < g_NumBooks; i++) {
        p = start;
        start = NULL;
        for (int j = 0; p->next->next != NULL; j++) {
          switch (method) {
                case 1:
                        more    =    strcmp(p->author, p->next->author);
                  if (more > 0) {
                        swapstr(p->author, p->next->author, g_tempStr); //використовуємо свій макрос свап
                        swapstr(p->name, p->next->name, g_tempStr);
                        swapint(p->year, p->next->year, tempint);
                        swapint(p->pages, p->next->pages, tempint);
                        swapint(p->cost, p->next->cost, tempint);
                  } break;
                case 2:
                         more    =    strcmp(p->name, p->next->name);
```

```
                if (more > 0) {
                        swapstr(p->author,
p->next->author, g_tempStr); //використовуємо
свій макрос свап
                        swapstr(p->name,
p->next->name, g_tempStr);
                        swapint(p->year,
p->next->year, tempint);
                        swapint(p->pages,
p->next->pages, tempint);
                        swapint(p->cost,
p->next->cost, tempint);
                } break;
              case 3:
                if (p->year > p->next->year) {
                        swapstr(p->author,
p->next->author, g_tempStr); //використовуємо
свій макрос свап
                        swapstr(p->name,
p->next->name, g_tempStr);
                        swapint(p->year,
p->next->year, tempint);
                        swapint(p->pages,
p->next->pages, tempint);
                        swapint(p->cost,
p->next->cost, tempint);
                } break;
              case 4:
                if (p->pages > p->next->pages) {
                        swapstr(p->author,
p->next->author, g_tempStr); //використовуємо
свій макрос свап
                        swapstr(p->name,
p->next->name, g_tempStr);
                        swapint(p->year,
p->next->year, tempint);
                        swapint(p->pages,
p->next->pages, tempint);
                        swapint(p->cost,
p->next->cost, tempint);
                } break;
              case 5:
                if (p->cost > p->next->cost) {
                        swapstr(p->author,
p->next->author, g_tempStr); //використовуємо
свій макрос свап
                        swapstr(p->name,
p->next->name, g_tempStr);
                        swapint(p->year,
p->next->year, tempint);
                        swapint(p->pages,
p->next->pages, tempint);
                        swapint(p->cost,
p->next->cost, tempint);
                } break;
          }
            if (!start) start = p; // після
сортування треба змінити
вказівник старт
          p = p->next;
        }
  }
}

// видалення книг за назвою
struct book *DeleteBook(struct book *p, struct book
*start)
{
  int tmp = 0; // для запису чи знайдено
книгу
  printf("Enter the name (or original part of name) of
the book that you want to delete\n");
  gets_s(g_tempStr);
        struct book *tmpStruct = (struct
book*)malloc(sizeof(struct book));
  for (int i = 0; p->next; i++) {
        if (p) tmpStruct = p;
        if (i != 0) p = tmpStruct->next;
        if (strstr(p->name, g_tempStr) != 0) {
          if (i == 0) start = start->next;
          tmpStruct->next = p->next;
          g_NumBooks--;
              printf("\n\tBook \"%s%\" deleted\n",
p->name);
          tmp = 1;
          free(p);       // одразу звільняє
пам*ять
          break;
        }
  }
  if (tmp == 0) printf("\n\tBook wasn't found\n\n");
  p = start;
  return start;
}

// пошук книг за назвою
// переміщає названу книгу вверх
int FindBook(struct book *p, struct book *start)
{
```

```
  int tempint = 0; // тимчасова змінна для свапання чисел
   printf("Enter the name (or original part of name) of the book \n");
  gets_s(g_tempStr);
  for (; p;) {
        if (strstr(p->name, g_tempStr) != 0) {
          swapstr(p->author, start->author, g_tempStr); //використовуємо свій макрос свап
          swapstr(p->name, start->name, g_tempStr);
          swapint(p->year, start->year, tempint);
          swapint(p->pages, start->pages, tempint);
          swapint(p->cost, start->cost, tempint);
          tempint = -1;
          break;
        }
        p = p->next;
  }
  if (tempint != -1) {    //        знайдено/не знайдено
        printf("\tBook not found\n");
        p = start;
        return 1;
   } else printf("\tBook \"%s\" found and took on the top\n", start->name);
  p = start;
  return 0;
}

// Зміна вибраної за назвою книги
int ChangeBook(struct book *p, struct book *start)
{
  if (FindBook(p, start) == 0) {   //          шукає функцією FindBook
        char answer;     // для відповіді чи треба далі змінювати книгу
        int method;      // запис що змінити в книзі
        printf("\n\tEditing book \"%s\"\n", p->name);
        while (1) {
           printf("What do you want to change?\n1 - Author,\n2 - Name,\n3 - Year,\n4 - Pages,\n5 - Cost,\n");
          gets_s(g_tempStr);
          method = atoi(g_tempStr);
          switch (method) {
                case 1: printf("Enter new author\t");
                  gets_s(p->author);
                  break;
                case 2: printf("Enter new name\t");
                  strcpy(p->name, " ");
                  gets_s(g_tempStr);
                  strcat(p->name, g_tempStr);
                  break;
                case 3: printf("Enter new year\t");
                  gets_s(g_tempStr);
                  p->year = atoi(g_tempStr);
                  break;
                case 4: printf("Enter new pages\t");
                  gets_s(g_tempStr);
                  p->pages = atoi(g_tempStr);
                  break;
                case 5: printf("Enter new cost\t");
                  gets_s(g_tempStr);
                  p->cost = atoi(g_tempStr);
                  break;
          }
            printf("Do you want to continue changing \"%s\"? (Y/N)\n", p->name);
          if ((answer = _getch()) != 'y') break;    // чи треба далі змінювати книгу?
        }
        return 0;
  }
  else return 1; // це якщо не знайдено книги
}
```

Додаткове завдання - створити Mind Map, я вибрав як область гру “STALKER”:



Висновки

Під час виконання лабораторної роботи я зрозумів, що розробка ПЗ складається з двох етапів: проектування і кодування, під час розробки ПЗ використовуються раніше створені зразки, що значно полегшує процес. Я навчився проектувати роботу, створювати зразки блок-схем, кодувати за визначеним стандартом, який дозволяє підвищити швидкість написання, читабельність і зрозумілість коду. Також я дізнався що таке Mind Map і навчився створювати власну карту думок.