МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”;
Інститут ІКНІ
Кафедра ПЗ

ЗВІТ

До лабораторної роботи №2

На тему: “Документування етапів проектування та кодування програми”

З дисципліни: “Вступ до інженерії Програмного забезпечення”

Лектор:
доцент каф. ПЗ
Левус Є.В.
Виконав:
Морозов О.Р.
студент групи ПЗ-11
Прийняв:
доцент каф. ПЗ
Левус Є.В.

Львів - 2022

Тема: Документування етапів проектування та кодування програми.

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

Теоретичні відомості

1. Які завдання вирішуються на етапі проектування ПЗ?

На етапі проектування вирішується з яких частин буде складатись ПЗ, як буде працювати, та якою буде його структура та поведінка.

21.Що таке угорська нотація змінних? Навести три приклади.

Угорська нотація - це метод найменування змінних в програмування при якому до ідентифікатора змінної або функції додається префікс який вказує на тип змінної. Наприклад:

pList - вказівник, префікс p;

bLarger - булева змінна, префікс b;

iAmount - цілочисельна змінна, префікс i;

30.Як можна провести навігацію по коду?

У Microsoft Visual Studio 2022 зручно проводити навігацію за допомогою додатку ReSharper C++, також можна проводити навігацію за допомогою команд та панелі навігації.

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);

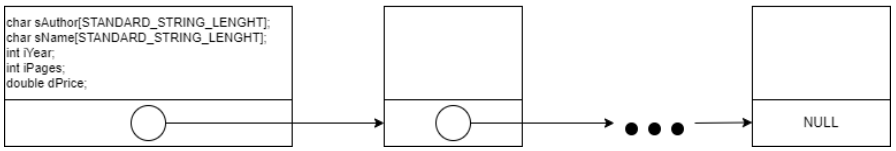
3. текст програми з коментарями та оформлений згідно вищенаведених рекомендацій щодо забезпечення читабельності й зрозумілості.

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

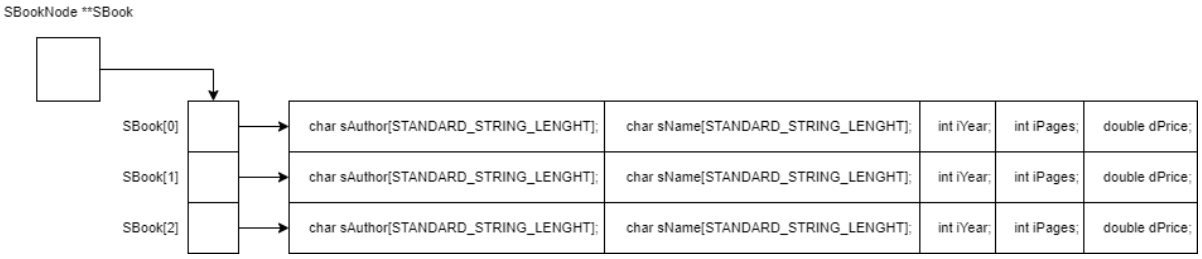
Отримані результати

1. Схематичне зображення структур даних:

   Однозв'язний список:

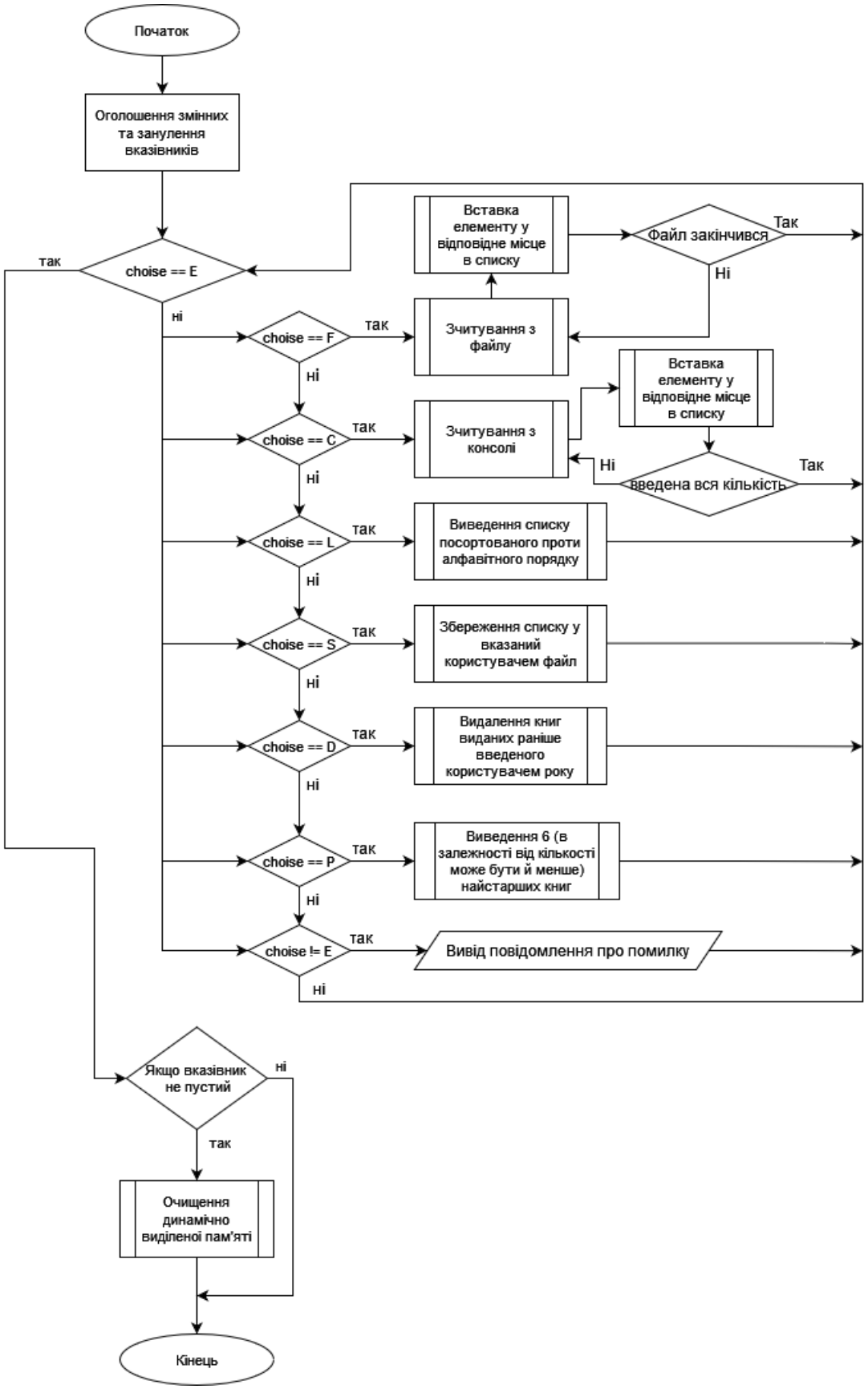


   Динамічне виділення пам'яті під структури:

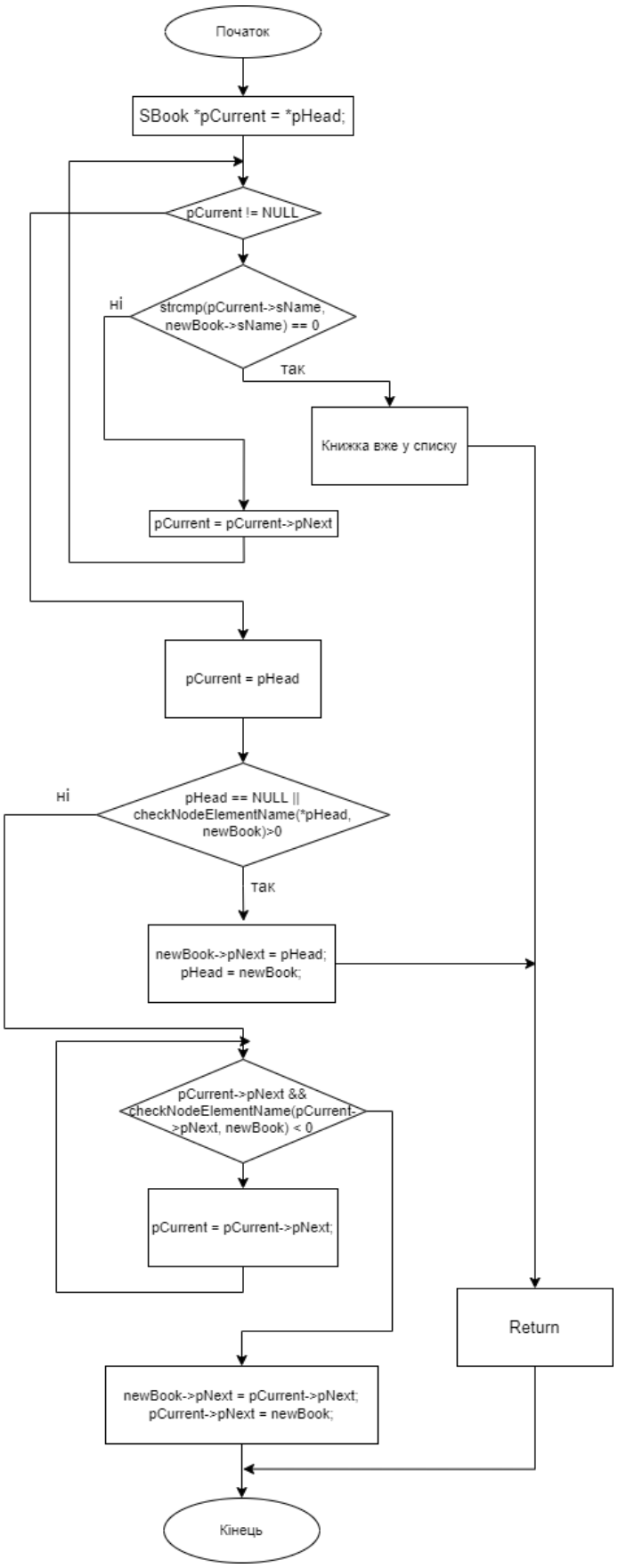


2. Блок-схеми алгоритмів ‑ основної функції та 2-ох окремих функцій‑підпрограм:

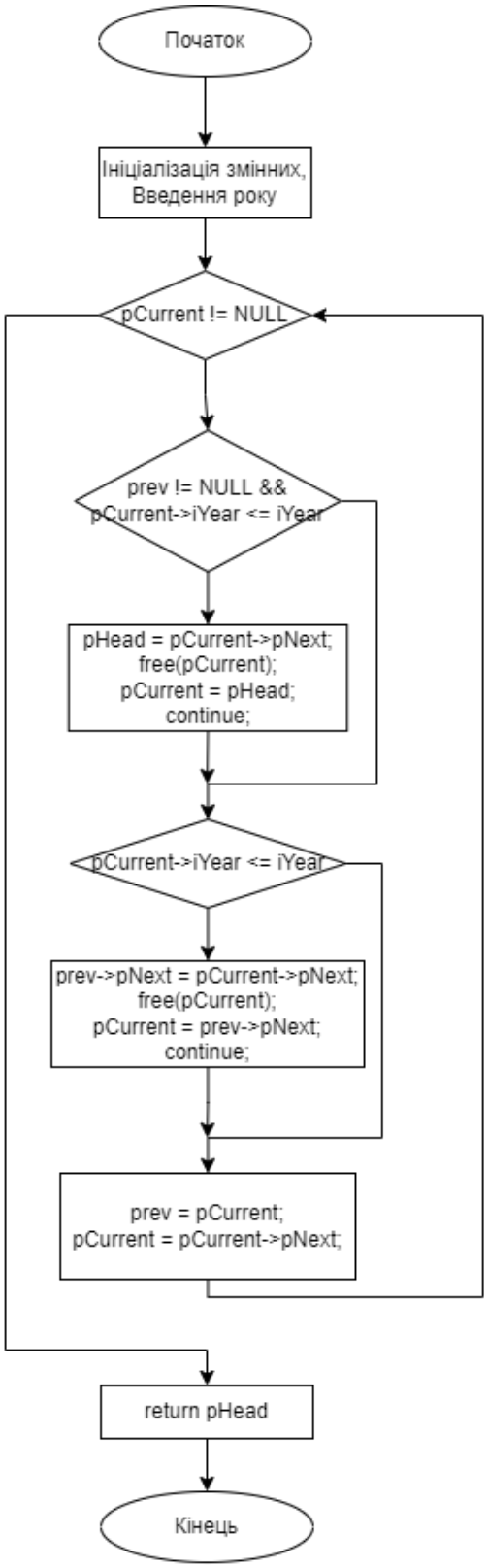
Блок-схема основної функції:



Блок-схема до алгоритму сортування методом вставки:


Початок
SBook *pCurrent = *pHead;
pCurrent != NULL
ні
strcmp(pCurrent->sName, newBook->sName) == 0
так
Книжка вже у списку
pCurrent = pCurrent->pNext
pCurrent = pHead
ні
pHead == NULL || checkNodeElementName(*pHead, newBook)>0
так
newBook->pNext = pHead; pHead = newBook;
pCurrent->pNext && checkNodeElementName(pCurrent->pNext, newBook) < 0
pCurrent = pCurrent->pNext;
Return
newBook->pNext = pCurrent->pNext; pCurrent->pNext = newBook;
Кінець

Блок-схема до алгоритму видалення книг виданих до введеного користувачем року:



3. Текст програми:

Файл functions.h

```
#define _CRT_SECURE_NO_WARNINGS
#pragma once
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define  STANDARD_STRING_LENGHT 120 //

typedef struct SBookNode {
   char sAuthor[STANDARD_STRING_LENGHT];
   char sName[STANDARD_STRING_LENGHT];
   int iYear;
   int iPages;
   double dPrice;

   struct SBookNode *pNext;
}SBook;

#ifndef FUNCTIONS_H
#define FUNCTIONS_H

//Scan and return char sumbol
//Used in menu to check what you choose
char checkTask(char choise);

//Save list of books in user choosed file
void saveListInFile(SBook *pHead);

//Function allocate memory for new object
SBook *addNewBook();

//Scan information from user choosed file to program memory
void readFromFile(SBook **pHead, int *iCount);

//Scan information from console to program memory
void readFromConsole(SBook **pHead, int *iCount);
```

```
//Function set new element in correct position in list by name in the opposite order to the alphabet
bool sortNewNodeElementByName(SBook **pHead, SBook *newBook);

//Function delete book published before year which user entered
SBook *deleteBooksByYear(SBook *pHead, int *iCount);

//Show list in program window(console)
void printListToConsole(SBook *pHead);

//clean program memory after exiting program
void freeDunamicAllocatedMemory(SBook *pHead);

//print some eldest books
//if count of books >= 6 print 6 books
//else print all books
void findEldestBookInList(SBook *pHead, SBook **pEldest, int iCount);
#endif // FUNCTIONS_H
```

Файл function.c

```
#include "functions.h"

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

//Check what letter you enter
//Return it to main function
char checkTask(char choise) {
	scanf("%c", &choise) ? (choise) : (choise = '\n');
	return choise;
}

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

//Allocate memory for new node element
//Return pointer to new node element
SBook *addNewBook() {
	SBook *newBook = (SBook *)calloc(1, sizeof(SBook));
```

```
	return newBook;
}

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

//Restart input when it was not correct
void restartLine(void) {
	while (getchar() != '\n');
}

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

//Enter new node elements to program memory from console
void readFromConsole(SBook **pHead, int *iCount) {
	int amount;
	printf("What amount of books you want to input in list?  \n");
	while(!scanf("%d", &amount) || amount <= 0) {
		restartLine();
		printf("You entered wrong number of books\nTry again: ");
	}
	while (amount > 0) {
		SBook *newBook = addNewBook();

		puts("Enter Author of the book: ");
		getc(stdin) ? 1: 0;
		gets_s(newBook->sAuthor, STANDARD_STRING_LENGHT);
		puts("Enter book name: ");
		gets_s(newBook->sName, STANDARD_STRING_LENGHT);
		printf("Enter year of publishing: ");
		while ((!scanf("%d", (&newBook->iYear))) || (newBook->iYear > 2022 ||
newBook->iYear < 1600)) {
			restartLine();
			printf("You entered wrong year\nTry again: ");
		}
		printf("Enter number of pages: ");
		while (!scanf("%d", (&newBook->iPages)) || newBook->iPages <= 0) {
			restartLine();
			printf("You enterd wrong amount of pages\nTry again: ");
```

```
		}
		printf("Enter book price: ");
		while ((!scanf("%lf", (&newBook->dPrice))) || (newBook->dPrice) < 0) {
			restartLine();
			printf("You entered wrong price of book\nTry again: ");
		}
		sortNewNodeElementByName(pHead, newBook);
		amount--;
		(*iCount)++;
	}
	printf("Input data from console done succesful\n");
}

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

//Enter new node elements from file to program memory
void readFromFile(SBook **pHead, int *iCount) {

	char buffer[STANDARD_STRING_LENGHT];

	printf("Type name of file to import list from this file\n");
	char filesName[STANDARD_STRING_LENGHT] = "database";
	getc(stdin) ? 1:0;
	gets(filesName, STANDARD_STRING_LENGHT);

	char txt[] = ".txt";
	strcat(filesName, txt);

	FILE *file;
	if (!(file = fopen(filesName, "r"))) {
		printf("Import error\nMaybe you want to do something else?\n");
		return;
	}

	bool bIfFileIsNull = false, bBookAddedInList = false;
	while (fgets(buffer, STANDARD_STRING_LENGHT, file)) {
		bIfFileIsNull = true;
		SBook *newBook = addNewBook();
```

```
			strcpy(newBook->sAuthor, strtok(buffer, "\t"));
			strcpy(newBook->sName, strtok(NULL, "\t"));
			char *numberBuffer = strtok(NULL, "\t");
			newBook->iYear = atol(numberBuffer);
			numberBuffer = strtok(NULL, "\t");
			newBook->iPages = atol(numberBuffer);
			numberBuffer = strtok(NULL, "\t");
			newBook->dPrice = atol(numberBuffer);

			bBookAddedInList = sortNewNodeElementByName(pHead, newBook);
			if (bBookAddedInList) {
				(*iCount)++;
			}
		}

		fclose(file);

		if (bIfFileIsNull) {
			printf("Import data from file %s done succesful\n", filesName);
		} else {
			printf("File %s is empty\n", filesName);
		}
}

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

//Sort by entering node element in the opposite order to the alphabet
bool sortNewNodeElementByName(SBook **pHead, SBook *newBook)  {
		SBook *pCurrent = *pHead;

		while (pCurrent) {
			if (strcmp(pCurrent->sName, newBook->sName) == 0) {
				printf("Book already in list\n");
				return false;
			}
			pCurrent = pCurrent->pNext;
		}
```

```
	pCurrent = *pHead;

	if (!*pHead || checkNodeElementName(pCurrent, newBook) > 0) {
		newBook->pNext = *pHead;
		*pHead = newBook;
		return true;
	}

	while (pCurrent->pNext && checkNodeElementName(pCurrent->pNext, newBook)
< 0) {
		pCurrent = pCurrent->pNext;//set pointer to correct position
	}
	newBook->pNext = pCurrent->pNext;//set node element to correct position
	pCurrent->pNext = newBook;
	return true;
}

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

//Check where new node element was comparing with the other elements in list
//Return 0 if name 1 and 2 books are equal
//<0 if the first non-matching character in book name 1 is greater (in ASCII) than that of
book name 2
//>0 if the first non-matching character in book name 1 is lower (in ASCII) than that of
book name 2.
int checkNodeElementName(SBook *one, SBook *two) {
	return strcmp(two->sName, one->sName );
}

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

//Delete node element which year is less or same to enter
SBook* deleteBooksByYear(SBook *pHead, int *iCount) {
	int iYear;
	printf("Enter year, book published before this year would be Delete from list\n");
	while (!scanf("%i", &iYear)) {
		restartLine();
		printf("Year is numbers, not letters or symbols\nTry again: ");
```

```
	}
	SBook *pCurrent = pHead;
	SBook *prev = NULL;
	while (pCurrent) {
		if ((!prev) && (pCurrent->iYear < iYear)) {
			pHead = pCurrent->pNext;
			free(pCurrent);
			pCurrent = pHead;
			(*iCount)--;
			continue;
		}

		if (pCurrent->iYear < iYear) {
			prev->pNext = pCurrent->pNext;
			free(pCurrent);
			pCurrent = prev->pNext;
			(*iCount)--;
			continue;
		}
		prev = pCurrent;
		pCurrent = pCurrent->pNext;
	}
	printf("Books published before %i year deleted\n", iYear);
	return pHead;
}

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

//Print list from program memory to console
void printListToConsole(SBook *pHead) {
	if (!pHead) {
		printf("Book not found\n");
		return;
	}

printf("-----------------------------------------------------------------------------------------------\n");
	printf("\t Author\t	|\t\t\t Name\t\t\t	| Published| Pages |	Price	|");
```

```
printf("\n-----------------------------------------------------------------------------------------------------\n")
;

	SBook *pCurrent = pHead;
	do {
		printf(" %-20s | %-44s|   %4d   |  %3d  |   %4.2lf   |\n",\
			pCurrent->sAuthor, pCurrent->sName, pCurrent->iYear,
pCurrent->iPages, pCurrent->dPrice);

		pCurrent = pCurrent->pNext;
	} while (pCurrent);


printf("-----------------------------------------------------------------------------------------------------\n\n")
;
}

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

//Clear memory after exiting program
void freeDunamicAllocatedMemory(SBook *pCurrent) {
	if (!pCurrent) {
		return;
	}
	freeDunamicAllocatedMemory(pCurrent->pNext);
	free(pCurrent);
}

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

//Save in file list from program memory
void saveListInFile(SBook *pHead) {
	if (!pHead) {
		printf("Book not found\n");
		return;
	}
	printf("Type name of file where you want to save\n");
```

```
	char filesName[STANDARD_STRING_LENGHT] = "filesName";
	scanf_s("%s", filesName, STANDARD_STRING_LENGHT);

	char txt[] = ".txt";
	strcat(filesName, txt);

	FILE *file;
	if (!(file = fopen(filesName, "a"))) {
		printf("Save error\nFile not open.\nReturn to menu\n");
		return;
	}
	SBook *pCurrent = pHead;
	do {
		fprintf(file, "%s\t%s\t%d\t%d\t%lf\n", \
			pCurrent->sAuthor, pCurrent->sName, pCurrent->iYear,
pCurrent->iPages, pCurrent->dPrice);

		pCurrent = pCurrent->pNext;
	} while (pCurrent);
	fclose(file);
	printf("Save data in file %s done succesful\n", filesName);
	return;
}

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

//Find 6 or less eldest books in list
void findEldestBookInList(SBook *pHead,SBook **pEldest,int iCount) {
	if (!pHead) {
		printf("Book not found\n");
		return;
	}

	for (int i = 0; i < iCount; i++) {
		SBook *pCurr = pHead;
		while (pCurr) {
			if (!pEldest[i]) {
				pEldest[i] = pCurr;
```

```
                        }else if (pEldest[i]->iYear < pCurr->iYear) {
                                bool iUsed = 0;
                                for (int j = 0; j < 6; j++) {
                                        if (pCurr == pEldest[j]) {
                                                iUsed = 1;
                                        }
                                }
                                if (!iUsed) {
                                        pEldest[i] = pCurr;
                                }
                        }
                        pCurr = pCurr->pNext;
                }
        }

        printf("---------------------------------------Eldest Books In
    List-----------------------------------------\n");
    printf("-------------------------------------------------------------------------------------------------\n");
        printf("\t Author\t     |\t\t\t Name\t\t\t    | Published| Pages |    Price   |");
    printf("\n-------------------------------------------------------------------------------------------------\n")
    ;
        for (int i = 0; i < iCount; i++) {
                printf(" %-20s | %-44s|   %4d   |  %3d  |   %4.2lf   |\n", \
                        pEldest[i]->sAuthor, pEldest[i]->sName, pEldest[i]->iYear,
    pEldest[i]->iPages, pEldest[i]->dPrice);
        }

printf("-------------------------------------------------------------------------------------------------\n\n");
    }
    //---------------------------------------------------------------------
```

Файл lab10.c

```
#include "functions.h"

int main(void) {

   SBook **pHead = NULL;
```

```
	int iCount = 0;

	char choise = '\0';
	printf("\t\t\t Menu\n");
	printf("Import from file - F\t\t Input from console - C\n");
	printf("Show list - L\t\t\t Save list in file - S\n");
	printf("Delete book by year - D\t Print 6 eldest books - P\nExit - E\n");

	while ((choise = checkTask(choise)) != 'E') {
		switch (choise) {
			case 'F': {
				readFromFile(&pHead, &iCount);//зчитування з файлу

				printf("\t\t\t Menu\n");
				printf("Import from file - F\t\t Input from console - C\n");
				printf("Show list - L\t\t\t Save list in file - S\n");
				printf("Delete book by year - D\t Print 6 eldest books - P\nExit - E\n");
				break;
			}
			case 'C': {
				readFromConsole(&pHead, &iCount);//зчитування з консолі

				printf("\t\t\t Menu\n");
				printf("Import from file - F\t\t Input from console - C\n");
				printf("Show list - L\t\t\t Save list in file - S\n");
				printf("Delete book by year - D\t Print 6 eldest books - P\nExit - E\n");
				break;
			}
			case 'L': {
				printListToConsole(pHead);//перегляд списку

				printf("\t\t\t Menu\n");
				printf("Import from file - F\t\t Input from console - C\n");
				printf("Show list - L\t\t\t Save list in file - S\n");
				printf("Delete book by year - D\t Print 6 eldest books - P\nExit - E\n");
```

```
                        break;
                }
                case 'S': {
                        saveListInFile(pHead);//збереження списку в файл

                        printf("\t\t\t Menu\n");
                        printf("Import from file - F\t\t Input from console - C\n");
                        printf("Show list - L\t\t\t Save list in file - S\n");
                        printf("Delete book by year - D\t Print 6 eldest books - P\nExit -
E\n");
                        break;
                }
                case 'D': {
                        pHead = deleteBooksByYear(pHead, &iCount);//видалення книг
молодше певного року

                        printf("\t\t\t Menu\n");
                        printf("Import from file - F\t\t Input from console - C\n");
                        printf("Show list - L\t\t\t Save list in file - S\n");
                        printf("Delete book by year - D\t Print 6 eldest books - P\nExit -
E\n");
                        break;
                }
                case 'P': {
                        int iEldestBooksCount = (iCount > 6) ? 6 : iCount;
                        SBook** pEldest = calloc(iEldestBooksCount, sizeof(SBook));
                        findEldestBookInList(pHead, pEldest, iEldestBooksCount);

                        printf("\t\t\t Menu\n");
                        printf("Import from file - F\t\t Input from console - C\n");
                        printf("Show list - L\t\t\t Save list in file - S\n");
                        printf("Delete book by year - D\t Print 6 eldest books - P\nExit -
E\n");
                        free(pEldest);
                        break;
                }
                default: {
                        if (choise == '\n') {
```

```
                                }
                                else {
                                        printf("Symbol not correct, look on Menu and try another\n");
                                        printf("\t\t\t Menu\n");
                                        printf("Import from file - F\t\t Input from console - C\n");
                                        printf("Show list - L\t\t\t Save list in file - S\n");
                                        printf("Delete book by year - D\t Print 6 eldest books - P\nExit - E\n");
                                }
                                break;
                        }
                }
        }

   if (pHead) {
                freeDunamicAllocatedMemory(pHead);
   }
        return 0;
}
```

Висновки

Виконуючи цю лабораторну роботу я навчився документувати основні етапи проектування та кодування ПЗ та засвоїв знання реалізувавши це на практиці.