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:

студ. групи ПЗ-15

Хробак О.М.

Прийняв:

викладач каф. ПЗ

Самбір А. А.

«___» _____ 2022 р.

∑ = _____

Львів – 2022.

Тема: документування етапів проектування та кодування програми
Мета: навчитися документувати основні результати етапів проектування та кодування найпростіших програм.

Теоретичні відомості

3. Що розуміють під архітектурою програмної системи? Архітектура програмної системи – це опис підсистем і компонентів програми, а також зв'язків між ними.

11. Які є властивості алгоритмів? Кожен алгоритм повинен володіти такими характеристиками:

1. Зрозумілість (виконавець повинен уміти виконувати кожну вказівку алгоритму, тобто розуміти кожну його команду);

2. Однозначність (всі дії алгоритму мають трактуватись однозначно);

3. Скінченність (алгоритм повинен приводити до результату за скінченну кількість дій);

4. Дискретність (поділ алгоритму на послідовність дій);

5. Універсальність (можливість застосовувати алгоритм для однотипних задач).

26. Які правила до найменування констант? Навести три приклади.

Назви мають бути великими літерами і розділятися '_'. Наприклад:

NUMBER_OF_ELEMENTS, SIZE_OF_ARRAY, MARK_COUNT.

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;

3. Текст програми з коментарями та оформлений згідно вище наведених рекомендацій щодо забезпечення читабельності й зрозумілості.

Умова до програми:

З текстового файлу зчитати послідовність записів, які містять дані про результати сесії студентів групи у такому форматі: <Прізвище> <Ім'я> <Дата народження> <Список Екзаменаційних оцінок>. Реалізувати операцію вставки нового елемента у відсортований список і операцію вилучення зі списку даних, які відповідають одній з наступних умов: про студентів, які отримали на першому та третьому іспитах оцінки 3;

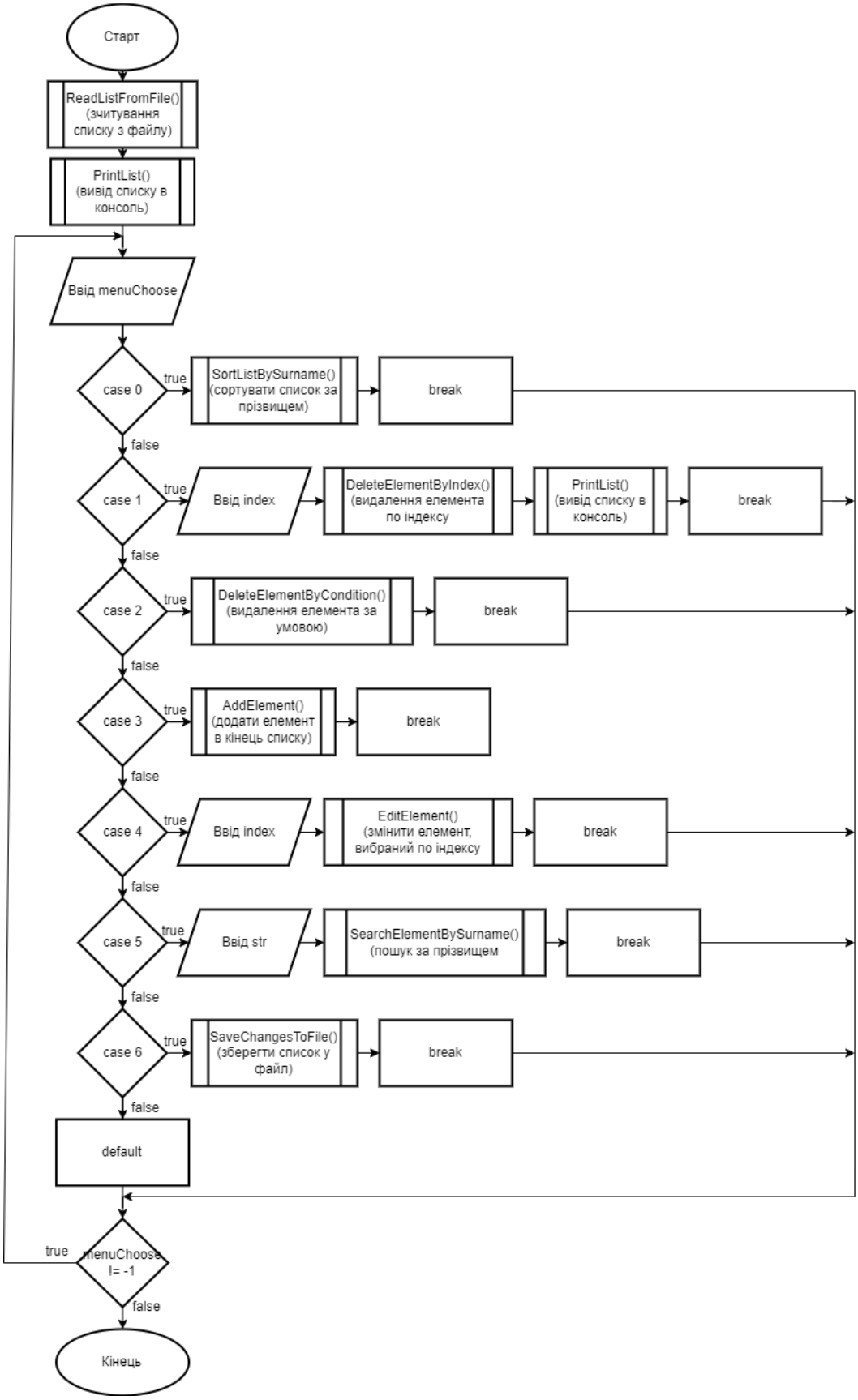
Хід роботи

1. Схематичне зображення структур даних, які використовуються для збереження інформації;

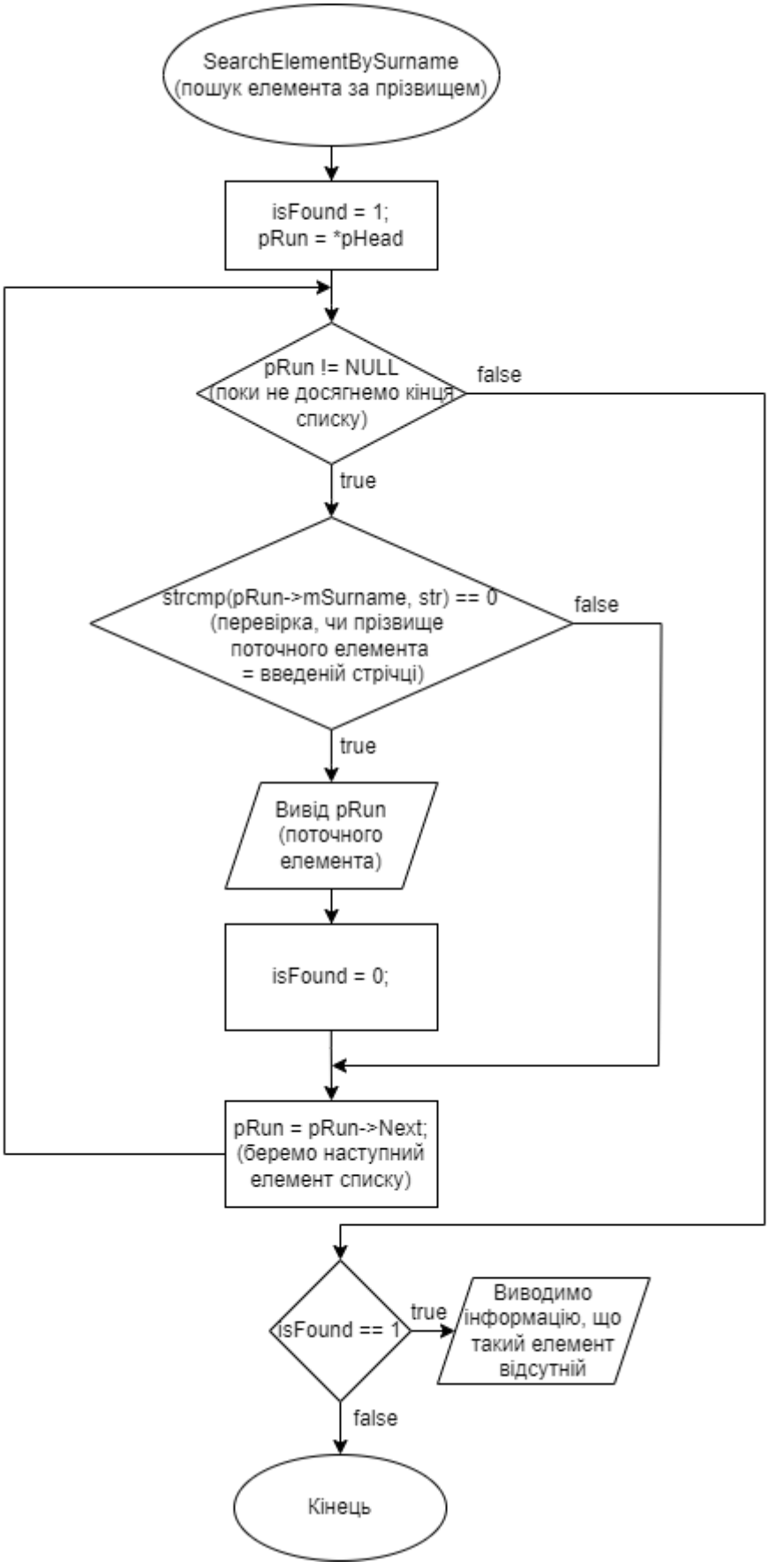
Student

`int mID;`
`int mMarks[MARKS_SIZE];`
`char mName[LENGTH];`
`char mSurname[LENGTH];`
`char mBirth[LENGTH];`

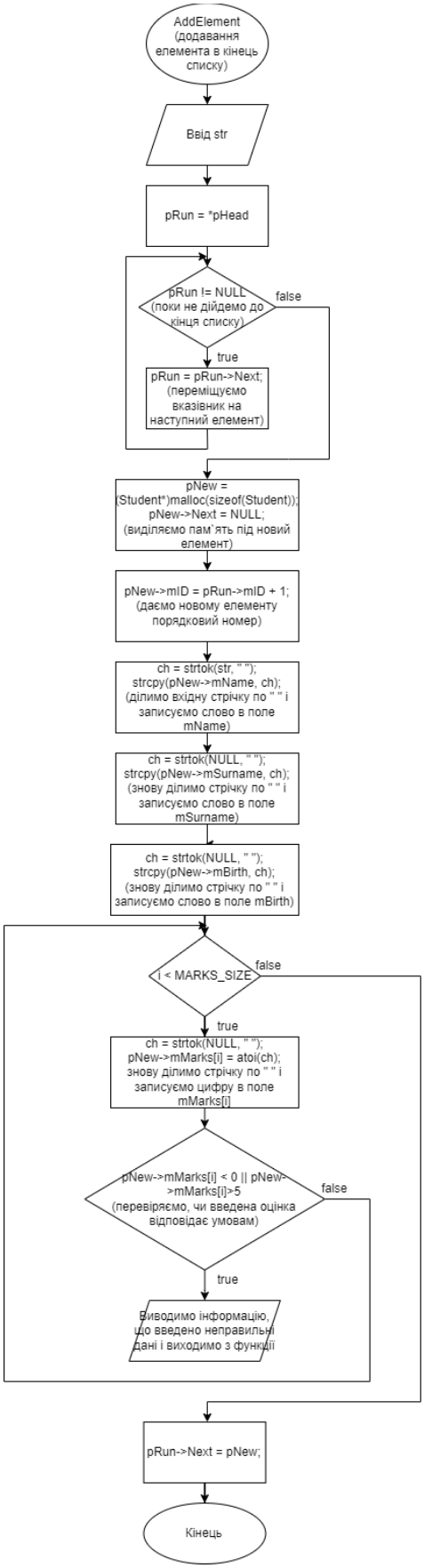
2. Блок-схема основної функції:



Блок-схема функції пошуку:



Блок-схема функції додавання елемента:



3. Код програми:

//FunctionsOfList.c

```
#define _CRT_SECURE_NO_WARNINGS

#include "FunctionsOfList.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 50



void ReadListFromFile(Student** pHead)  // Read elements from binary file and
fill the list with them.
{
	Student* pRun = NULL, * pNew = NULL, *pCurr = NULL;
	int i = 1;
	FILE* pFile;

	if ((pFile = fopen("students.bin", "rb")) == NULL)
	{
		printf("Can`t find a file.");
		return;
	}


	while (!feof(pFile))
	{
		if ((*pHead) == NULL)
		{
			(*pHead) = (Student*)malloc(sizeof(Student));
			(*pHead)->Next = NULL;
			fread((*pHead), sizeof(Student), 1, pFile);
			(*pHead)->mID = i;
			pRun = *pHead;
		}

		i++;
		pNew = (Student*)malloc(sizeof(Student));
		pNew->Next = NULL;

		fread(pNew, sizeof(Student), 1, pFile);
		pNew->mID = i;

		pRun->Next = pNew;
		pRun = pNew;

	}
	pRun = *pHead;
	while (pRun->Next)
	{
		pCurr = pRun;
		pRun = pRun->Next;
	}
	if (pRun)
	{
		pCurr->Next = pRun->Next;
		free(pRun);
	}
```

```
}

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

void PrintList(const Student** pHead) // Displays list in the console.
{
	Student* pStart = *pHead;
	printf("           Student           |  Date of birth  |       Marks
|\n");
	printf("-----------------------------------------------------------------
--\n");
	while (pStart)
	{
		printf("%2d.%14s %11s|%17s|", pStart->mID, pStart->mName, pStart-
>mSurname, pStart->mBirth);
		printf("    ");

		for (int j = 0; j < MARKS_SIZE; j++)
		{
			printf("%d; ", pStart->mMarks[j]);
		}
		printf("|\n");
		pStart = pStart->Next;
	}
	printf("-----------------------------------------------------------------
--\n");
}

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

void SortListBySurname(Student** pHead) // Sort list alphabetically by the
surname.
{
	printf("\033[0d\033[2J");
	Student* pRun = NULL, pTemp, *pHead2 = NULL;

	pHead2 = *pHead;

	if (*pHead)
	{
		while (pHead2->Next)
		{
			pRun = pHead2->Next;
			do
			{
				if (strcmp(pRun->mSurname, pHead2->mSurname) < 0)
				{
					strcpy(pTemp.mName, pRun->mName);
					strcpy(pRun->mName, pHead2->mName);
					strcpy(pHead2->mName, pTemp.mName);

					strcpy(pTemp.mSurname, pRun->mSurname);
					strcpy(pRun->mSurname, pHead2->mSurname);
					strcpy(pHead2->mSurname, pTemp.mSurname);

					strcpy(pTemp.mBirth, pRun->mBirth);
					strcpy(pRun->mBirth, pHead2->mBirth);
					strcpy(pHead2->mBirth, pTemp.mBirth);

					for (int i = 0; i < MARKS_SIZE; i++)
					{
						pTemp.mMarks[i] = pRun->mMarks[i];
						pRun->mMarks[i] = pHead2->mMarks[i];
```

```
                                        pHead2->mMarks[i] = pTemp.mMarks[i];
                                }
                        }
                        pRun = pRun->Next;
                } while (pRun);
                pHead2 = pHead2->Next;
        }

        printf("List was successfully sorted:\n\n");
        PrintList(pHead);
    }
}

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

void DeleteHead(Student** pHead) // Delete the head(first element) of the list
{
    Student* pCurr = NULL, * pRun = NULL, * pTemp = NULL, * pCurr2 = NULL;
    pRun = *pHead;
    int isDeleted = pRun->mID;

    (*pHead) = (*pHead)->Next;
    free(pRun);
    pRun = NULL;

    pTemp = *pHead;
    while (pTemp)
    {
        pTemp->mID = pTemp->mID - 1;
        pCurr2 = pTemp;
        pTemp = pTemp->Next;
    }
}

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

void setNewIndexes(Student** pHead, int isDeleted) // Set new indexes for
remaining elements of the list after deleting an element.
{
    Student* pCurr = NULL, * pRun = NULL;
    pRun = *pHead;
    while ((pRun) && (pRun->mID != (isDeleted + 1)))
    {
        pCurr = pRun;
        pRun = pRun->Next;
    }
    while (pRun)
    {
        pRun->mID = pRun->mID - 1;
        pCurr = pRun;
        pRun = pRun->Next;
    }
}

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

void DeleteElementByCondition(Student** pHead) // Delete elements for which the
delete condition is valid (those who has mark "3" for the first and third exam)
{
    if (*pHead == NULL)
    {
        printf("List is empty!");
        return;
```

```
	}

	Student* pCurr = NULL, * pRun = NULL, * pTemp = NULL, * pCurr2;
	pRun = *pHead;
	char str[8] = "";
	int delElem = 0;
	char del[2] = "";
	int isDeleted = 0;
	Student* pDel = NULL;

	pRun = *pHead;
	while (pRun)
	{
		if ((*pHead)->mMarks[0] == 3 && (*pHead)->mMarks[2] == 3)
		{
			DeleteHead(pHead);
			continue;
		}

		if ((pRun) && (pRun->mMarks[0] == 3) && (pRun->mMarks[2] == 3))
		{
			isDeleted = pRun->mID;
			pCurr->Next = pRun->Next;
			pDel = pRun->Next;
			free(pRun);
			pRun = pDel;

			setNewIndexes(pHead, isDeleted);

			continue;
		}
		pCurr = pRun;
		pRun = pRun->Next;
	}
	printf("\033[0d\033[2J");

	printf("Elements were successfully deleted.\n\n");
	PrintList(pHead);
	return;

}

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

void DeleteElementByIndex(Student** pHead, int index) // Delete an element by the
index set by the user.
{
	if (*pHead == NULL)
	{
		printf("List is empty!");
		return;
	}

	Student* pCurr = NULL, * pRun = NULL, * pTemp = NULL, * pCurr2, * pDel =
NULL;
	pRun = *pHead;
	int delElem = index;
	int isDeleted = 0;

	if ((*pHead)->mID == delElem)
	{
		DeleteHead(pHead);
	}
```

```
	else
	{
		while ((pRun) && (pRun->mID != delElem) )
		{
			pCurr = pRun;
			pRun = pRun->Next;
		}
		if (pRun)
		{
			isDeleted = pRun->mID;
			pCurr->Next = pRun->Next;
			free(pRun);
		}
		else
		{
			printf("There isn`t any element with this number\n");
			return;
		}

		setNewIndexes(pHead, isDeleted);
	}
	printf("\033[0d\033[2J");
	printf("Element was successfully deleted.\n\n");

}

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

void EditElement(Student** pHead, int index) // Edit an element with the index
set by the user.
{
	Student* pRun = *pHead;
	char str[SIZE] = "";
	char* ch = NULL;

	printf("\nEnter info(example: Mark Collins 21.12.2005 5 4 3 2 1):\n");
	while ((getchar()) != '\n');
	gets(str);

	while ((pRun) && (pRun->mID != index))
	{
		pRun = pRun->Next;
	}
	if (pRun)
	{
		ch = strtok(str, " ");
		strcpy(pRun->mName, ch);
		ch = strtok(NULL, " ");
		strcpy(pRun->mSurname, ch);
		ch = strtok(NULL, " ");
		strcpy(pRun->mBirth, ch);
		for (int i = 0; i < MARKS_SIZE; i++)
		{
			ch = strtok(NULL, " ");
			pRun->mMarks[i] = atoi(ch);
			if (pRun->mMarks[i] < 0 || pRun->mMarks[i]>5)
			{
				printf("Wrong input!");
				return -1;
			}
		}
		printf("\033[0d\033[2J");
		printf("Element was successfully edited.\n\n");
```

```
			PrintList(pHead);
		}
		else
		{
			printf("Invalid index!");
		}
}

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

void SearchElementBySurname(Student** pHead, char str[]) // Search an element by
the surname set by the user.
{
		Student* pRun = *pHead;
		int isFound = 1;
		while (pRun)
		{
			if (strcmp(pRun->mSurname, str) == 0)
			{
				isFound = 0;
				printf("Students found!\n%d. | %s | %s | %s | ", pRun->mID,
pRun->mName, pRun->mSurname, pRun->mBirth);

				for (int j = 0; j < MARKS_SIZE; j++)
				{
					printf("%d; ", pRun->mMarks[j]);
				}
				printf("|\n");
			}
			pRun = pRun->Next;
		}
		if (isFound)
		{
			printf("Can`t find any students with this surname!");
		}
}

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

void AddElement(Student** pHead) // Add a new element at the end of the list
{
		Student* pRun, *pNew;
		char str[SIZE] = "";
		char* ch = NULL;

		printf("\nEnter info(example: Mark Collins 21.12.2005 5 4 3 2 1):\n");
		while ((getchar()) != '\n');
		gets(str);

		pRun = *pHead;

		while (pRun->Next)
		{
			pRun = pRun->Next;
		}
		pNew = (Student*)malloc(sizeof(Student));
		pNew->Next = NULL;

		pNew->mID = pRun->mID + 1;
		ch = strtok(str, " ");
		strcpy(pNew->mName, ch);
		ch = strtok(NULL, " ");
		strcpy(pNew->mSurname, ch);
```

```
	ch = strtok(NULL, " ");
	strcpy(pNew->mBirth, ch);
	for (int i = 0; i < MARKS_SIZE; i++)
	{
		ch = strtok(NULL, " ");
		pNew->mMarks[i] = atoi(ch);
		if (pNew->mMarks[i] < 0 || pNew->mMarks[i]>5)
		{
			printf("Wrong input!");
			return -1;
		}
	}
	pRun->Next = pNew;

	printf("\033[0d\033[2J");
	printf("Element was successfully added.\n\n");
	PrintList(pHead);
}

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

void SaveChangesToFile(const Student** pHead) // Save current list to the binary
file.
{
	Student* pRun = NULL;
	FILE* pFile;

	pFile = fopen("students1.bin", "wb");

	pRun = (Student*)(*pHead);
	while (pRun)
	{
		fwrite(pRun, sizeof(Student), 1, pFile);
		pRun = pRun->Next;
	}

	printf("Your last changes was successfully saved.");
}
```

// FunctionsOfList.h

```
#define _CRT_SECURE_NO_WARNINGS
#ifndef FUNC_H
#define FUNC_H

#define SIZE 30
#define LENGTH 100
#define MARKS_SIZE 5


typedef struct Student
{
	int mID;
	int mMarks[MARKS_SIZE];
	char mName[LENGTH];
	char mSurname[LENGTH];
	char mBirth[LENGTH];
	struct Student* Next;

}Student;
```

```
void ReadListFromFile(Student** pHead);

void PrintList(const Student** pHead);

void SortListBySurname(Student** pHead);

void DeleteElementByCondition(Student** pHead);

void DeleteElementByIndex(Student** pHead, int index);

void AddElement(Student** pHead);

void EditElement(Student** pHead, int index);

void SearchElementBySurname(Student** pHead, char str[]);

void SaveChangesToFile(const Student** pHead);

#endif
```

// lab10.c

```
#define _CRT_SECURE_NO_WARNINGS
#include "FunctionsOfList.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
	Student* pHead = NULL;
	ReadListFromFile(&pHead);
	PrintList(&pHead);

	int menuChoose = 0;
	int index = 0;
	char del[2] = "";
	char str[SIZE] = "";

	do
	{
		printf("\n\n 0 - Sort list by surname.\n 1 - Delete element by
index.\n 2 - Delete element by condition (those who has mark \"3\" for the first
and third exam).");
		printf("\n 3 - Add new element to the end of list.\n 4 - Edit
element\n 5 - Search element by surname.\n 6 - Save changes.\n\n\n - 1 to
exit.\nChoose what you want to do: ");
		scanf("%d", &menuChoose);
		switch (menuChoose)
		{
		case 0:
			SortListBySurname(&pHead);
			break;
		case 1:
			printf("\n\nEnter number of element which you want to delete:
");
			scanf("%d", &index);
			DeleteElementByIndex(&pHead, index);
			PrintList(&pHead);
			break;
		case 2:
			DeleteElementByCondition(&pHead);
			break;
```

```
            case 3:
                  AddElement(&pHead);
                  break;
            case 4:
                  printf("\n\nEnter number of element which you want to edit:
");
                  scanf("%d", &index);
                  EditElement(&pHead, index);
                  break;
            case 5:
                  printf("\nEnter the surname of the student you want to
find:\n");
                  while ((getchar()) != '\n');
                  gets(str);
                  SearchElementBySurname(&pHead, str);
                  break;
            case 6:
                  SaveChangesToFile(&pHead);
                  break;
            default:
                  printf("\nWrong input!");
            }
      } while (menuChoose != -1);

      return 0;
}
```

Висновки

Під час виконання лабораторної роботи я опрацював другий та третій етапи ЖЦ ПЗ – проектування та кодування, та сформував пакет документів до розробленої раніше програми, а саме:

1. Схематичне зображення використаних структур даних
2. Блок-схеми алгоритмів
3. Код програми