НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЗВІТ

Про виконання лабораторної роботи № 2

з дисципліни «Вступ до інженерії програмного забезпечення»

Лекторка:

доцент Левус Є. В.

Виконала:

студ. групи ПЗ-16

Голомша О.Я.

Прийняв:

викладач 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 – опис підсистем і компонентів програмної системи, а також зв'язків між ними.

15. Скільки входів та виходів має блок розгалуження? Відповідь пояснити.

Блок розгалуження має один вхід та два виходи. Блок має два виходи, оскільки в залежності від виконання чи невиконання певної умови здійснюється або одна, або друга послідовність дій.

24. Як правильно розділяти функції у тексті програми?

Імплементація функцій розділяється стрічкою довжиною 80 символів:

//--

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 функційпідпрограм (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.

Отримані результати

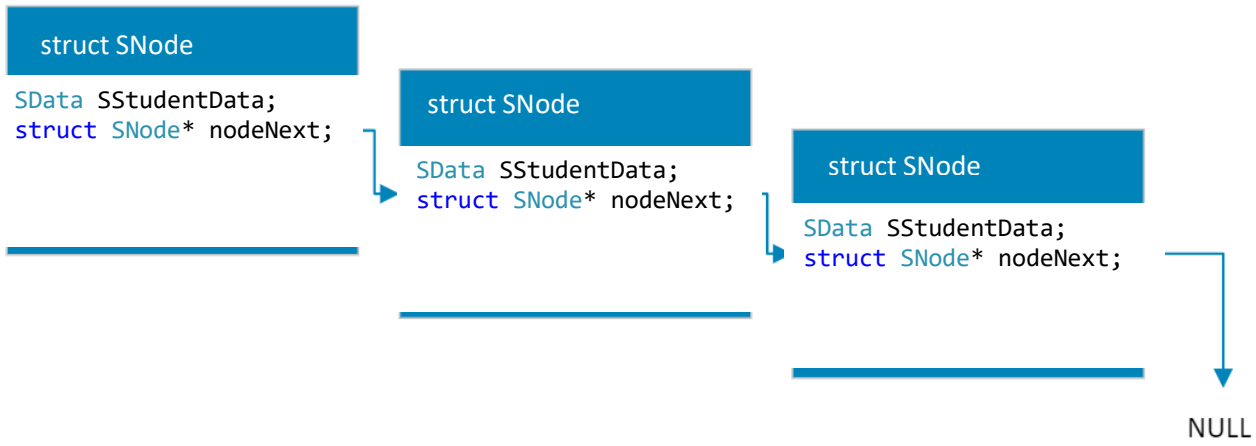


Рис. 1. Однозв’язний список.

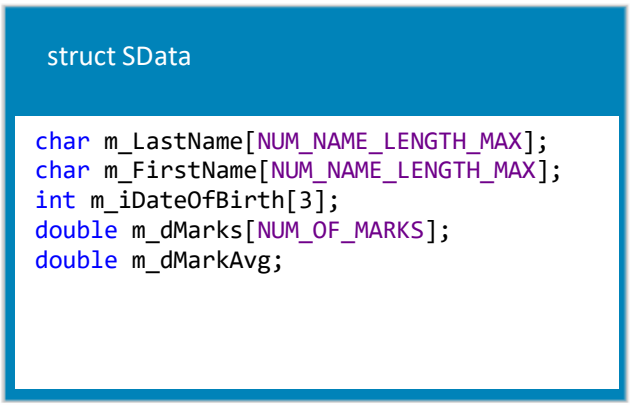
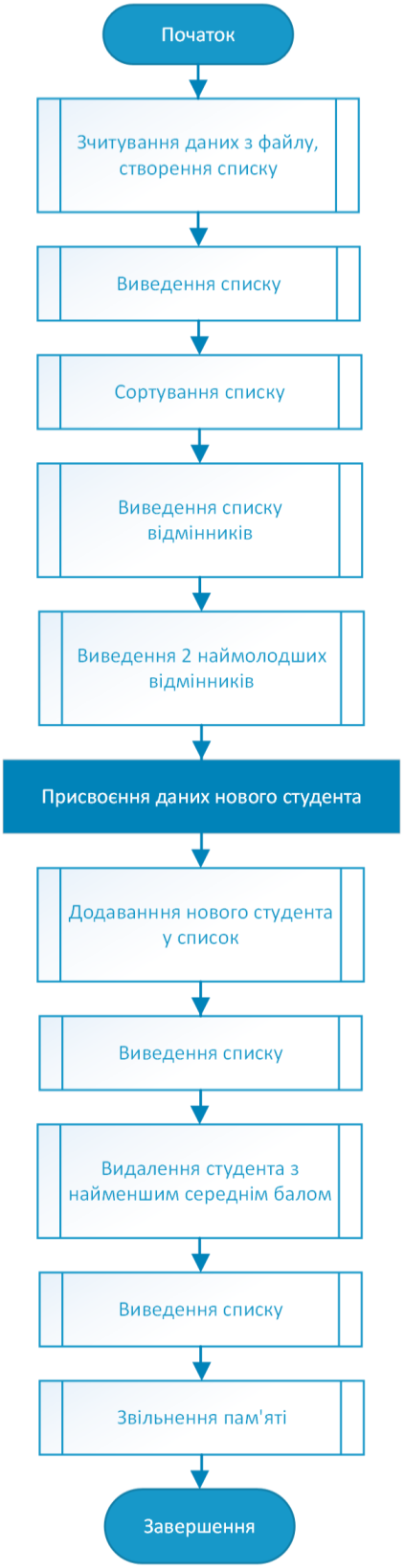
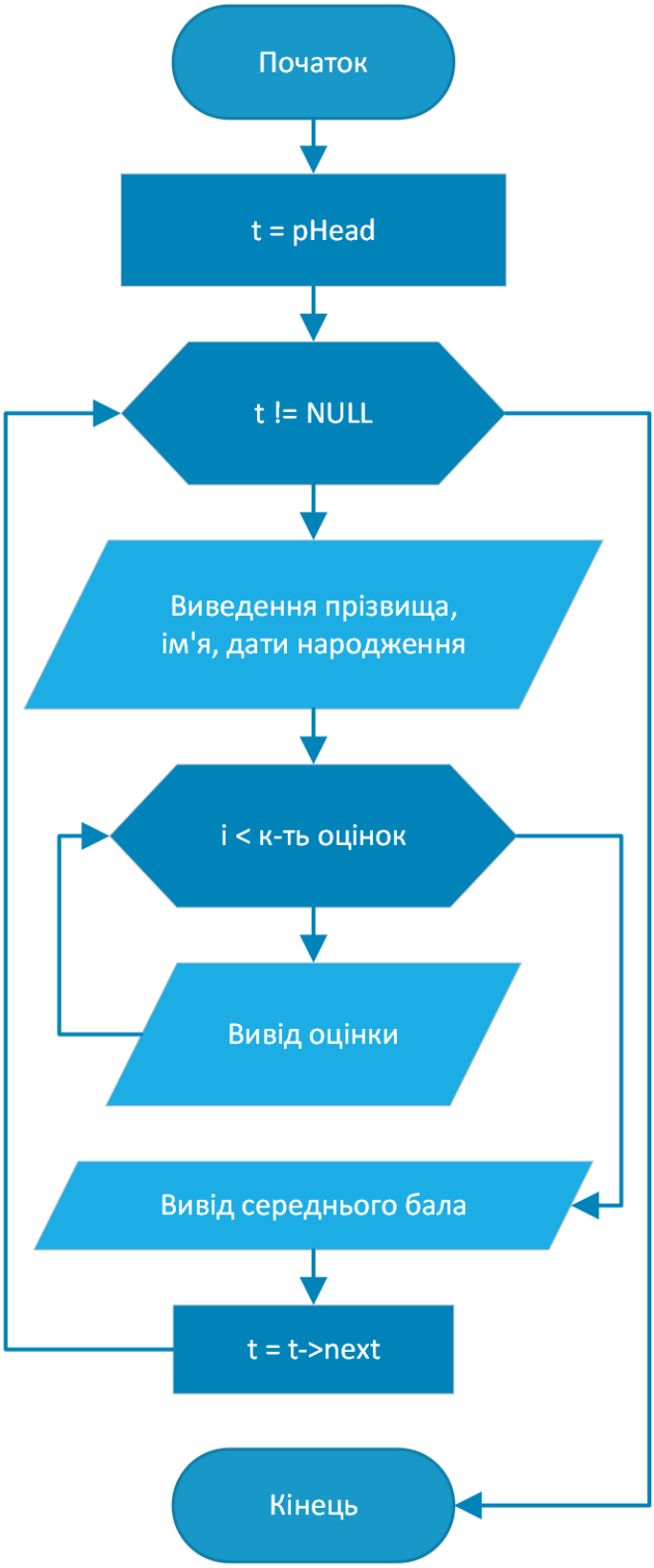


Рис. 2. Структура даних про студента

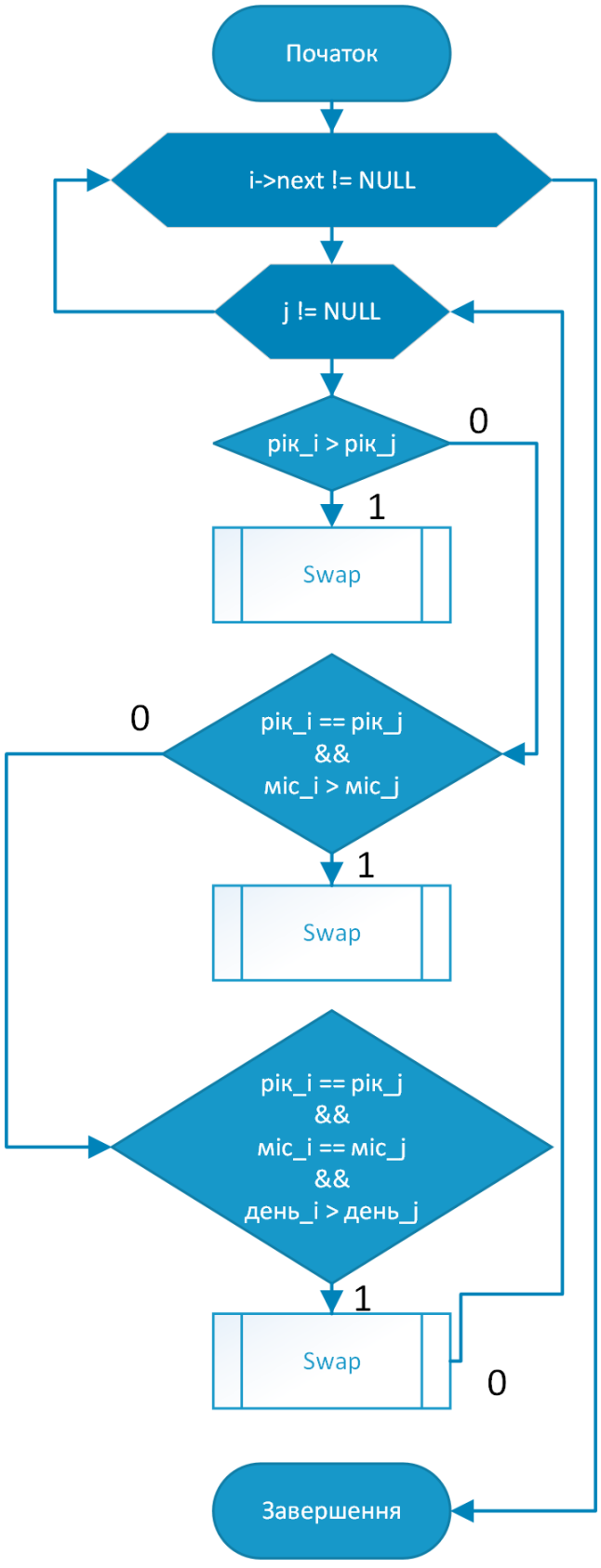
Функція main:



Функція PrintList:



Функція SortList:



КОД ПРОГРАМИ

Main.c

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#define NUM_NAME_LENGTH_MAX 20
#define NUM_OF_MARKS 5
#define NUM_OF_STUDENTS 10
#include "functions10.h"

int main() {
	FILE* pFileWithList;
	pFileWithList = fopen("pz16.txt", "r");
	if (pFileWithList == NULL) {
		printf("Can`t open file.");
	}

	SNode* pHead = NULL;
	pHead = createList(NUM_OF_STUDENTS, pFileWithList);
	printf("Initial data from the file.\n");
	printList(pHead);
	fclose(pFileWithList);

	printf("\nSorted data (by age).\n");
	sortList(&pHead);
	printList(pHead);

	printf("\nStudents with marks 4 or 5.\n");
	printListWithMarks4and5(pHead);

	printf("\nThe youngest students with marks 4 or 5.\n");
	print2TheYoungest(pHead);

	printf("\nList with an insertwd node.\n");
	SData newStudent = { "Holomsha", "Oksana", { 12, 6, 2003 }, { 4, 3, 4, 5, 3 }, 3.8 };
	addNodeToList(&pHead, newStudent);
	printList(pHead);

	printf("\nList with deleted nodes.\n");
	deleteNodesWithMinAvgMark(&pHead);
	printList(pHead);

	freeMemory(&pHead);
	return 0;

}
```

Functions10.h

```c
#ifndef FUNCTIONS10_H
#define FUNCTIONS10_H
#define NUM_NAME_LENGTH_MAX 20
#define NUM_OF_MARKS 5
typedef struct SData {
	char m_LastName[NUM_NAME_LENGTH_MAX];
	char m_FirstName[NUM_NAME_LENGTH_MAX];
```

```
	int m_iDateOfBirth[3];
	double m_dMarks[NUM_OF_MARKS];
	double m_dMarkAvg;
}SData;

typedef struct SNode {
	SData SStudentData;
	struct SNode* nodeNext;
}SNode;

// SNode* createList(int n, FILE* pFile)
//
//Summary of the createtList function:
//		The function creates the singly-linked list from the file.
//
// Parameters:
//		Integer number of students in the input file.
//		Pointer to the file with students.
//
// Return values:
//		Pointer to the head of the list.
//
// Description:
//		The function gets values from the file and creates a singly-linked list.
//		Note! The avarage mark is counted autimatically.
//
//		Reference of data in file:
//		Loo Lopiue	12.12.1212	1 2 3 4 5
//		Loo		name (left alignment)
//		Lopiue		surname (left alignment)
//		12.12.1212	date of birth
//		1 2 3 4 5	the int array of marks (Note! There is a space " " between each
mark.)
//
//	Note! There is a tabulation "	" between each column.
SNode* createList(int n, FILE* pFile);
//void printList(SNode* pHead)
//
//Summary of the printList function:
//		The function outputs the given list to the console.
//
// Parameters:
//		Pointer to the head of the list.
//
// Return values:
//		Nothing.
//
// Description:
//		The function prins the list to the console like that:
//		Loo Lopiue	12.12.1212	1 2 3 4 5	3
//
//		Loo		name (left alignment)
//		Lopiue		surname (left alignment)
//		12.12.1212	date of birth
//		1 2 3 4 5	the int array of marks (Note! There is a space " " between each
mark.)
//		3		avarage mark
//
//	Note! There is a tabulation "	" between each column.
void printList(SNode*  pHead);
//void printListWithMarks4and5(SNode* pHead)
```

```
//
//Summary of the printListWithMarks4and5 function:
//            The function outputs only students with avarage marks 4 and 5
//            from the given list to the console.
//
// Parameters:
// Return values:          same as for void printList(SNode*  pHead)
// Description:
void printListWithMarks4and5(SNode* pHead);
//void print2TheYoungest(SNode* pHead)
//
//Summary of the print2TheYoungest function:
//            The function outputs only 2 the youngest students with avarage marks 4 and 5
//            from the given list to the console.
//
// Parameters:
// Return values:          same as for void printList(SNode*  pHead)
// Description:
void print2TheYoungest(SNode* pHead);
//void sortList(SNode** ppHead)
//
//Summary of the sortList function:
//            The function sorts data of the list, from older students to younger.
//
// Parameters:
//            Pointer to pinter to the head of the list.
//
// Return values:
//            Nothing.      Note! Modifies the list.
//
// Description:
//            This function utilizes the standard selection sort algorithm.
//            Note! The list is modified via swapping data.
void sortList(SNode** ppHead);
//void addNodeToList(SNode** ppHead, SData newStudentData)
//
//Summary of the addNodeToList function:
//            The function adds data about new student into it`s correct place in the sorted
list.
//
// Parameters:
//            Pointer to pointer to the head of the list.
//            Struct with info for new student.
//
// Return values:
//            Nothing.      Note! Modifies the list.
//
// Description:
//            This function adds new node to the list and than sorta the list.
//            Note! The list is modified via adding node.
void addNodeToList(SNode** ppHead, SData newStudentData);
//void deleteNodesWithMinAvgMark(SNode** ppHead)
//
//Summary of the deleteNodesWithMinAvgMark function:
//            The function deletes node with minimal avarage mark from the list.
//
// Parameters:
//            Pointer to pointer to the head of the list.
//
// Return values:
//            Nothing.      Note! Modifies the list.
```

```
//
// Description:
//            This function deletes node with min avg mark.
//            Note! The list is modified via deleting node.
void deleteNodesWithMinAvgMark(SNode** ppHead);
//void deleteNode(SNode** ppHead, int position)
//
//Summary of the deleteNode function:
//            The function deletes node from the certain position of the list.
//
// Parameters:
//            Pointer to pinter to the head of the list.
//            Integer position of the node.
//
// Return values:
//            Nothing.      Note! Modifies the list.
//
// Description:
//            This function deletes node from the certain position of the list.
//            Note! The list is modified via deleting node.
void deleteNode(SNode** ppHead, int position);
//void freeMemory(SNode** ppHead)
//
//Summary of the freeMemory function:
//            The function cleans dynamically allocated memory from the heap.
//
// Parameters:
//            Pointer to pinter to the head of the list.
//
// Return values:
//            Nothing.
//
// Description:
//            This function deletes all the data from the list and cleans the memory.
void freeMemory(SNode** ppHead);

#endif
```

Functions10.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include "functions10.h"
#define NUM_NAME_LENGTH_MAX 20
#define NUM_OF_MARKS 5

SNode* createList(int n, FILE* pFile) {
	SNode* pnodeHead = NULL;
	SNode* pnodeTemp = NULL;
	SNode* p = NULL;

	for (int i = 0; i < n; i++) {
		pnodeTemp = (SNode*)malloc(sizeof(SNode));
		for (int i = 0; i < 3; ++i) {
			fscanf(pFile, "%d.", &(pnodeTemp->SStudentData.m_iDateOfBirth[i]));
		}
		double sum = 0;
```

```
		for (int j = 0; j < NUM_OF_MARKS; j++) {
			fscanf(pFile, "%lf, ", &(pnodeTemp->SStudentData.m_dMarks[j]));
			sum += pnodeTemp->SStudentData.m_dMarks[j];
		}
		sum /= NUM_OF_MARKS;
		pnodeTemp->SStudentData.m_dMarkAvg = sum;
		pnodeTemp->nodeNext = NULL;

		if (pnodeHead == NULL)
			pnodeHead = pnodeTemp;
		else {
			p = pnodeHead;
			while (p->nodeNext != NULL)
				p = p->nodeNext;
			p->nodeNext = pnodeTemp;
		}
	}
	return pnodeHead;
}
//-------------------------------------------------------------------------------
void printList(SNode* pHead) {
	SNode* t = pHead;
	while (t != NULL) {
		printf("%-11s %-10s\t", t->SStudentData.m_LastName, t->SStudentData.m_FirstName);
		printf("%d.%d.%d\t", t->SStudentData.m_iDateOfBirth[0], t-
>SStudentData.m_iDateOfBirth[1], t->SStudentData.m_iDateOfBirth[2]);
		for (int i = 0; i < NUM_OF_MARKS; i++) {
			printf("%.0lf ", t->SStudentData.m_dMarks[i]);
		}
		printf("\t%.1lf\n", t->SStudentData.m_dMarkAvg);
		t = t->nodeNext;
	}
}
//-------------------------------------------------------------------------------
void printListWithMarks4and5(SNode* pHead) {
	SNode* t = pHead;
	while (t != NULL) {
		int r = 0;
		for (int i = 0; i < NUM_OF_MARKS; i++) {
			if (t->SStudentData.m_dMarks[i] < 4)
				r = 1;
			else continue;
		}
		if (r == 0) {
			printf("%-11s %-10s\t", t->SStudentData.m_LastName, t-
>SStudentData.m_FirstName);
			printf("%d.%d.%d\t", t->SStudentData.m_iDateOfBirth[0], t-
>SStudentData.m_iDateOfBirth[1], t->SStudentData.m_iDateOfBirth[2]);
			for (int i = 0; i < NUM_OF_MARKS; i++) {
				printf("%.0lf ", t->SStudentData.m_dMarks[i]);
			}
			printf("\t%.1lf\n", t->SStudentData.m_dMarkAvg);
		}
		t = t->nodeNext;
	}
}
//-------------------------------------------------------------------------------
void print2TheYoungest(SNode* pHead) {
	SNode* t = pHead;
	int count = 0;
	while (t != NULL) {
```

```
		int r = 0;
		for (int i = 0; i < NUM_OF_MARKS; i++) {
			if (t->SStudentData.m_dMarks[i] < 4)
				r = 1;
			else continue;
		}
		if (r == 0 && count > 2) {
			printf("%-11s %-10s\t", t->SStudentData.m_LastName, t-
>SStudentData.m_FirstName);
			printf("%d.%d.%d\t", t->SStudentData.m_iDateOfBirth[0], t-
>SStudentData.m_iDateOfBirth[1], t->SStudentData.m_iDateOfBirth[2]);
			for (int i = 0; i < NUM_OF_MARKS; i++) {
				printf("%.0lf ", t->SStudentData.m_dMarks[i]);
			}
			printf("\t%.1lf\n", t->SStudentData.m_dMarkAvg);
		}
		count++;
		t = t->nodeNext;
	}
}
//---------------------------------------------------------------------------
void sortList(SNode** ppHead) {
	SNode* i, * j;
	for (i = *ppHead; i->nodeNext != NULL; i = i->nodeNext) {
		for (j = i->nodeNext; j != NULL; j = j->nodeNext) {
			if (i->SStudentData.m_iDateOfBirth[2] > j->SStudentData.m_iDateOfBirth[2])
{
				SData temp = i->SStudentData;
				i->SStudentData = j->SStudentData;
				j->SStudentData = temp;
			}
			else if (i->SStudentData.m_iDateOfBirth[2] == j-
>SStudentData.m_iDateOfBirth[2] &&
					i->SStudentData.m_iDateOfBirth[1] > j-
>SStudentData.m_iDateOfBirth[1]) {
				SData temp = i->SStudentData;
				i->SStudentData = j->SStudentData;
				j->SStudentData = temp;
			}
			else if (i->SStudentData.m_iDateOfBirth[2] == j-
>SStudentData.m_iDateOfBirth[2] &&
					i->SStudentData.m_iDateOfBirth[1] == j-
>SStudentData.m_iDateOfBirth[1] &&
					i->SStudentData.m_iDateOfBirth[0] > j-
>SStudentData.m_iDateOfBirth[0]) {
				SData temp = i->SStudentData;
				i->SStudentData = j->SStudentData;
				j->SStudentData = temp;
			}
		}
	}
}
//---------------------------------------------------------------------------
void addNodeToList(SNode** ppHead, SData newStudentData) {
	SNode* pnew = (SNode*)malloc(sizeof(SNode));
	pnew->SStudentData = newStudentData;
	pnew->nodeNext = NULL;
	pnew->nodeNext = *ppHead;
	*ppHead = pnew;
	sortList(ppHead);
}
```

```
//-----------------------------------------------------------------------------
void deleteNodesWithMinAvgMark(SNode** ppHead) {
	SNode* i = *ppHead;
	double min = i->SStudentData.m_dMarkAvg;
	int min_pos = 0;
	int pos = 0;
	for (i = *ppHead; i != NULL; i = i->nodeNext) {
		if (i->SStudentData.m_dMarkAvg < min) {
			min = i->SStudentData.m_dMarkAvg;
			pos = min_pos;
		}
		min_pos++;
	}
	deleteNode(ppHead, pos);

	i = *ppHead;
	min = i->SStudentData.m_dMarkAvg;
	min_pos = 0;
	pos = 0;
	for (i = *ppHead; i != NULL; i = i->nodeNext) {
		if (i->SStudentData.m_dMarkAvg < min) {
			min = i->SStudentData.m_dMarkAvg;
			pos = min_pos;
		}
		min_pos++;
	}
	deleteNode(ppHead, pos);

	i = *ppHead;
	min = i->SStudentData.m_dMarkAvg;
	min_pos = 0;
	pos = 0;
	for (i = *ppHead; i != NULL; i = i->nodeNext) {
		if (i->SStudentData.m_dMarkAvg < min) {
			min = i->SStudentData.m_dMarkAvg;
			pos = min_pos;
		}
		min_pos++;
	}
	deleteNode(ppHead, pos);
}
//-----------------------------------------------------------------------------
void deleteNode(SNode** ppHead, int pos) {
	if (*ppHead == NULL)
		return;

	SNode* temp = *ppHead;

	if (pos == 0) {
		*ppHead = temp->nodeNext;
		free(temp);
		return;
	}

	for (int i = 0; temp != NULL && i < pos - 1; i++)
		temp = temp->nodeNext;


	if (temp == NULL || temp->nodeNext == NULL)
		return;
```

```
	SNode* next = temp->nodeNext->nodeNext;

	free(temp->nodeNext);
	temp->nodeNext = next;
}
//-------------------------------------------------------------------------
void freeMemory(SNode** ppHead) {
	SNode* temp = *ppHead;
	SNode* elem = NULL;
	while (temp != NULL) {
		elem = temp;
		temp = temp->nodeNext;
		free(elem);
	}

}
```

Висновки

Під час виконання лабораторної роботи я навчилася документувати основні результати етапів проектування та кодування найпростіших програм.