МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут комп'ютерних наук та інформаційних технологій

Кафедра програмного забезпечення

ЗВІТ

До лабораторної роботи № 2

З дисципліни: "Вступ до інженерії програмного забезпечення"

Лектор:

Левус Є. В.

Виконала:

ст. гр. ПЗ-15

Пшеницька Н. А.

Прийняв:

Самбір А. А.

Львів – 2022

Тема роботи: документування етапів проектування та кодування програми.

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

Теоретичні відомості

8. Порівняти переваги/недоліки у застосуванні масивів чи однозв'язних списків для власної програми.

- Переваги застосування однозв'язних списків: швидка вставка/видалення елемента, зручні для зберігання даних
- Недоліки застосування однозв'язних списків: довго доступатися до потрібного елемента списку, займає доволі багато пам'яті

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

- Якщо функція виконує дію, то краще використовувати дієслово, ніж іменник для зрозумілості коду (наприклад, CheckPrompt() замість PromptLine()).
- Використовувати загальновикористовувані префікси (Is - для перевірки, Get - повернення значення, Set - встановлення значення)
- Якщо функція виконує дію над об'єктом, то треба використовувати дієслово + іменник (наприклад, SortList()).
- Не використовувати назви класів для методів цього ж класу (замість Student.StudentData краще Student.Data)
- Використовувати суфікси: Desc - порядок спадання, Asc - порядок зростання, Avg - середнє значення

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

- Статичні методи, конструктори, деструктори, змінні-члени, оператори, методи-члени

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.

Отримані результати

1. Схематичне зображення структур даних:

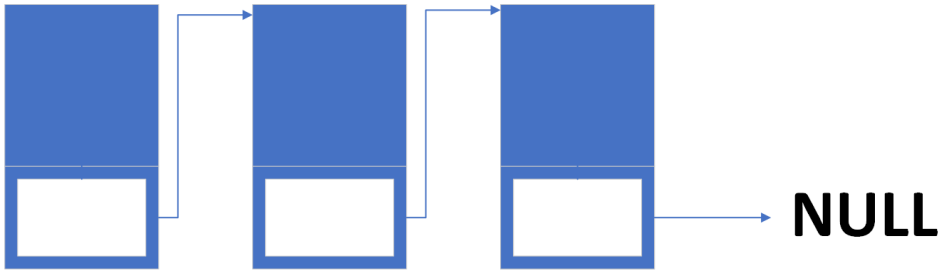
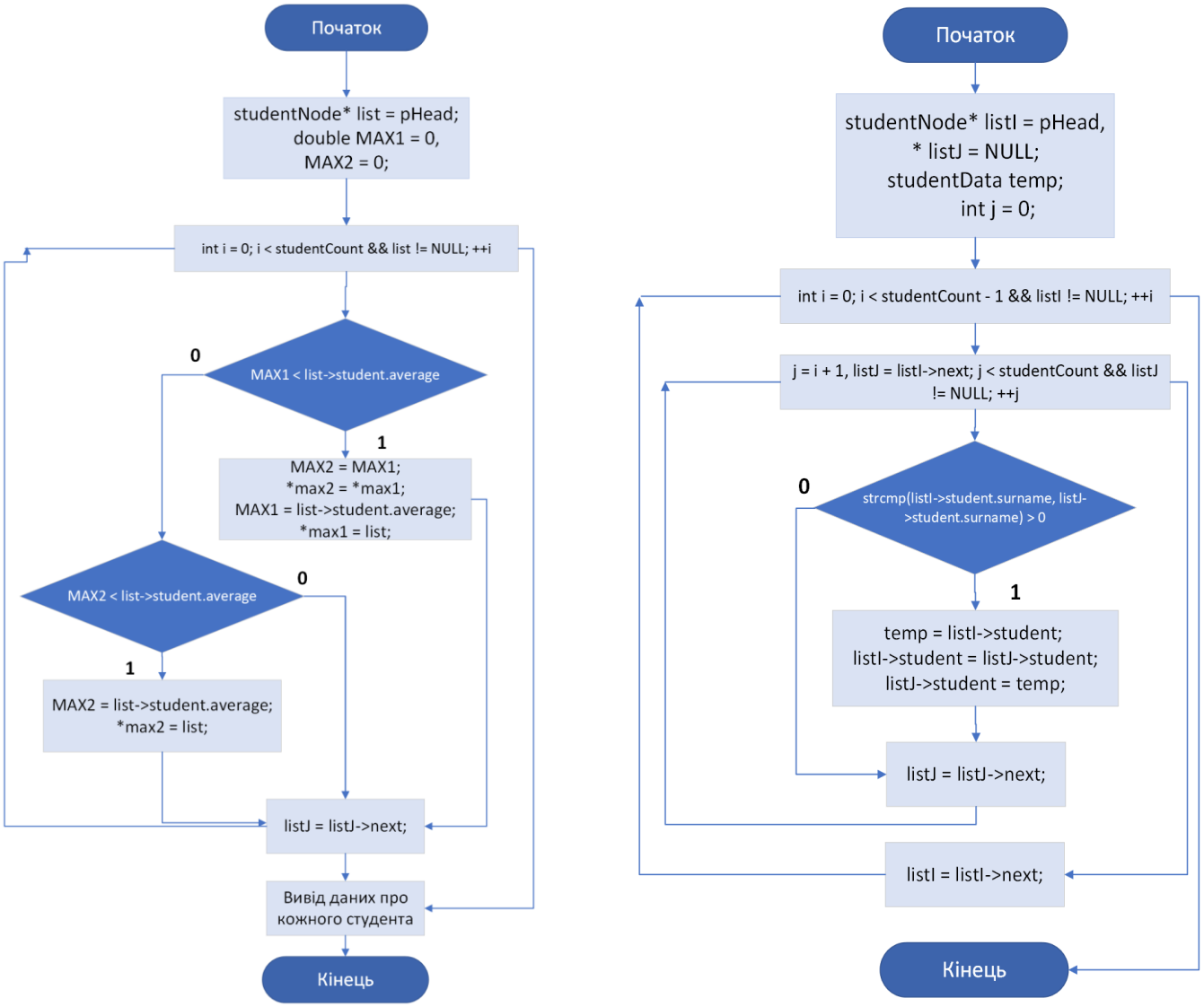


Рис. 1 Однозв'язний список

Рис. 2 Одновимірний масив

2. Блок-схеми алгоритмів:

Рис. 3 Блок-схема алгоритму основної функції



а) *б)*

Рис. 4 Блок-схеми алгоритмів функцій:
а) знаходження двох студентів з максимальним середнім балом
б) сортування списку в алфавітному порядку

3. Текст програми:

Файл *Header1.h*

```
#ifndef HEADER1_H
#define HEADER1_H

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

#define STR 200
#define SIZE 31
#define NUM_OF_GRADES 4
#define DOC "C:\\Users\\Nadia\\Desktop\\new.txt"
#define NAME "NAME"
```

```
#define SURNAME "SURNAME"
#define DATE "DATE OF BIRTH"
#define GRADES "GRADES"

typedef struct student
{
	char surname[SIZE];
	char name[SIZE];
	char dateOfBirth[SIZE];
	int grades[NUM_OF_GRADES];
	double average;

} studentData;
typedef struct student_node
{
	studentData student;
	struct student_node* next;
} studentNode;

void CheckPrompt(int argc, char* argv[]);
int FromFileToList();
void PrintList();
void SortList();
void TwoStudMaxAverage(studentNode** max1, studentNode** max2);
void DeleteStudsWithoutFive();
void FreeList();
#endif //HEADER1_H
```

Файл *functions.c*

```
#include "Header1.h"

studentNode* pHead;
int studentCount, NUM;

//receive and check arguments from prompt line about number of students
void CheckPrompt(int argc, char* argv[]) {
	if (argc == 2) {
		NUM = atoi(argv[1]);
		if (!NUM) {
			printf("COUDN'T READ FROM COMMAND PROMT. PLEASE ENTER DATA HERE ");
			scanf("%d", &NUM);
		}
	}
	else {
		printf("COUDN'T READ FROM COMMAND PROMT. PLEASE ENTER DATA HERE");
		scanf("%d", &NUM);
	}
}

//read the file and write data to list
int FromFileToList() {
	studentNode* pStud, * last = NULL;
	char strOneStudent[STR], * pCharTemp;
	double sumGrade = 0;
	FILE* fListOfStudents = fopen(DOC, "r");
	if (!fListOfStudents) {
		printf("COUDN'T OPEN AND READ FROM FILE\n");
		return 0;
	}

	while (!feof(fListOfStudents)) {
		fgets(strOneStudent, STR, fListOfStudents);

		pStud = (studentNode*)malloc(sizeof(studentNode));
```

```
			if (pStud) {
				pCharTemp = strtok(strOneStudent, " \t\n,");
				strcpy(pStud->student.surname, pCharTemp);
				pCharTemp = strtok(NULL, " \t\n,");
				strcpy(pStud->student.name, pCharTemp);
				pCharTemp = strtok(NULL, " \t\n,");
				strcpy(pStud->student.dateOfBirth, pCharTemp);
				sumGrade = 0;
				for (int i = 0; i < NUM_OF_GRADES; ++i) {
					pCharTemp = strtok(NULL, " \t\n,");
					pStud->student.grades[i] = atoi(pCharTemp);
					sumGrade += (double)pStud->student.grades[i];
				}
				pStud->student.average = sumGrade / NUM_OF_GRADES;

				if (!pHead) {
					pHead = pStud;
					last = pHead;
				}
				else {
					last->next = pStud;
					last = pStud;
				}

				pStud->next = NULL;
				++studentCount;
				if (NUM == studentCount) break;
			}
			else return 0;
		}
	}

	//print given list
	void PrintList()
	{
		if (!pHead) {
			printf("\nCOUDND FIND YOUR LIST OF STUDENTS.\n");
			return;
		}
		studentNode* pList = pHead, * pel = pHead;
		int maxNameLen = 0, maxSurnameLen = 0, maxDateLen = 0, temp = 0, generalLen = 0;

		while (SURNAME[temp++] != '\0') ++maxSurnameLen;
		temp = 0;
		while (NAME[temp++] != '\0') ++maxNameLen;
		temp = 0;
		while (DATE[temp++] != '\0') ++maxDateLen;

		int nameN = 0, surnameN = 0, dateN = 0;
		for (int i = 0; i < studentCount && pList != NULL; ++i)
		{
			temp = 0;
			nameN = 0, surnameN = 0, dateN = 0;
			while (pList->student.surname[temp++] != '\0') ++surnameN;
			temp = 0;
			while (pList->student.name[temp++] != '\0') ++nameN;
			temp = 0;
			while (pList->student.dateOfBirth[temp++] != '\0')
				++dateN;

			if (surnameN > maxSurnameLen) maxSurnameLen = surnameN;
			if (nameN > maxNameLen) maxNameLen = nameN;
			if (dateN > maxDateLen)   maxDateLen = dateN;
			pList = pList->next;
		}
```

```
	generalLen = maxNameLen + maxSurnameLen + maxDateLen + 38;
	temp = 0;
	while (temp++ < generalLen) printf("-");
	printf("\n| %-*s | %-*s | %-*s | %-16s| %-8s|\n", maxSurnameLen, SURNAME,
maxNameLen, NAME, maxDateLen, DATE, GRADES, "AVERAGE");

	temp = 0;
	while (temp++ < generalLen)
		printf("-");

	while (pel != NULL) {
		printf("\n| %-*s | %-*s | %-*s | ", maxSurnameLen, pel->student.surname,
maxNameLen, pel->student.name, maxDateLen, pel->student.dateOfBirth);
		for (int i = 0; i < NUM_OF_GRADES; ++i)
			printf("%-3d ", pel->student.grades[i]);
		printf("| %-7.3lf |", pel->student.average);
		pel = pel->next;
	}
	printf("\n");
	temp = 0;
	while (temp++ < generalLen) printf("-");
}

//sort list in ascending order by name
void SortList() {
	studentNode* listI = pHead, * listJ = NULL;
	studentData temp;
	int j = 0;
	for (int i = 0; i < studentCount - 1 && listI != NULL; ++i) {
		for (j = i + 1, listJ = listI->next; j < studentCount && listJ != NULL;
++j) {
			if (strcmp(listI->student.surname, listJ->student.surname) > 0) {
				temp = listI->student;
				listI->student = listJ->student;
				listJ->student = temp;
			}
			listJ = listJ->next;
		}
		listI = listI->next;
	}
}

//find and print data about two students with max average grade
void TwoStudMaxAverage(studentNode** max1, studentNode** max2) {
	studentNode* list = pHead;
	double MAX1 = 0, MAX2 = 0;
	studentData dataMax1, dataMax2;
	for (int i = 0; i < studentCount && list != NULL; ++i) {
		if (MAX1 < list->student.average) {
			MAX2 = MAX1;
			*max2 = *max1;
			MAX1 = list->student.average;
			*max1 = list;
		}
		else if (MAX2 < list->student.average) {
			MAX2 = list->student.average;
			*max2 = list;
		}
		list = list->next;
	}
	dataMax1 = (**max1).student;
	dataMax2 = (**max2).student;

	printf("\n\nSTUDENTS WITH THE HIGHEST AVERAGE GRADE\n\n");

	for (int j = 0; j < 70; ++j) printf("-");
```

```
	printf("\n");
	printf("| %-12s| %-11s| %-13s| %-15s| %-8s|\n", SURNAME, NAME, DATE, GRADES,
"AVERAGE");
	for (int j = 0; j < 70; ++j) printf("-");
	printf("\n");

	printf("| %-12s| %-11s| %-13s", dataMax1.surname, dataMax1.name,
dataMax1.dateOfBirth);
	for (int i = 0; i < NUM_OF_GRADES; ++i) printf("|%3d", dataMax1.grades[i]);
	printf(" | %8.4lf|\n", dataMax1.average);


	printf("| %-12s| %-11s| %-13s", dataMax2.surname, dataMax2.name,
dataMax2.dateOfBirth);
	for (int i = 0; i < NUM_OF_GRADES; ++i) printf("|%3d", dataMax2.grades[i]);
	printf(" | %8.4lf|\n", dataMax2.average);

	for (int j = 0; j < 70; ++j) printf("-");
	printf("\n");
}

//delete students who doesnt have mark 5
void DeleteStudsWithoutFive() {
	int isFive = 0;
	studentNode* start = pHead, * prev = start;
	for (int i = 0; i < studentCount; ++i) {
		isFive = 0;
		for (int j = 0; j < NUM_OF_GRADES; ++j) {
			if (start->student.grades[j] >= 88)
				isFive = 1;
		}
		if (!isFive) {
			if (start == pHead) {
				pHead = start->next;
				free(start);
				start = pHead;
			}
			else {
				prev->next = start->next;
				free(start);
				start = prev->next;
			}
		}
		else {
			prev = start;
			start = start->next;
		}
	}
}

//free dynamic memory used for list and delete all list
void FreeList() {
	studentNode* temp = pHead;
	while (pHead) {
		temp = pHead->next;
		free(pHead);
		pHead = temp;
	}
     }
```

Файл *main.c*

```
#include "Header1.h"

int main(int argc, char* argv[])
{
```

```
 studentNode* max1 = NULL, * max2 = NULL;

 CheckPrompt(argc, argv);
 FromFileToList();
 PrintList();
 SortList();
 printf("\n\nAFTER SORTING\n\n");
 PrintList();
 TwoStudMaxAverage(&max1, &max2);
 DeleteStudsWithoutFive();
 printf("\n\nAFTER DELETING\n\n");
 PrintList();
 FreeList();
}
```

Висновок: у ході виконання даної лабораторної роботи я навчилася документувати основні результати етапів проектування та кодування найпростіших програм. Я схематично зобразила використані структури даних за допомогою редактора MS-Visio, розробила блок-схеми алгоритмів певних функцій програми, написала код програми відповідно до правил оформлення коду на мові C/C++.