Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення

­­



**Звіт**

Про виконання 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:**

ст. гр. ПЗ-11

Козира О.Ф.

**Прийняла:**

Доцент каф. ПЗ

Левус Є. В.

« \_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_\_ \_\_\_\_\_\_\_\_\_ .

Львів – 2022

**Тема.** Документування етапів проектування та кодування програми.

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

**Теоретичні відомості**

11. Які властивості алгоритмів?

- здатність до поділу на послідовність дій

- однозначність

- універсальність

- зрозумілість

- скінченність (результативність)

33. Що таке статичний аналіз коду?

Це аналіз коду, який починається одразу ж при відкритті відповідного файлу без виконання самої програми, після чого можна виявити деякі помилки або знайти фрагменти коду, які можна покращити. Наприклад, за допомогою статичного аналізу коду можна виявити чи існує опис прототипу функції.

25. Які правила до найменування змінних? Навести три приклади.

- назва кожної змінної повинна починатися її типом, наприклад int\* pnArray

- використовувати пари змінних min/max, begin/end, open/close

- додавати в кінець назви кваліфікатори Min, Max, Avg, Index при необхідності

**Завдання**

*Частина I.* У розробленій раніше програмі до лабораторної роботи з дисципліни «Основи програмування» внести зміни - привести 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 II.* 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 наведених рекомендацій щодо забезпечення читабельності й зрозумілості.

**Результат**

Схематичне зображення структур даних:

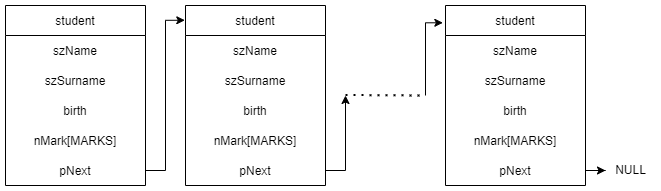


Рис.1. Однозв'язний список структур student

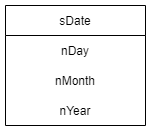


Рис.2. Структура sDate, що є полем birth структури student

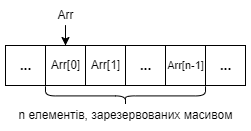


Рис.3. Одновимірний масив

Блок-схеми:

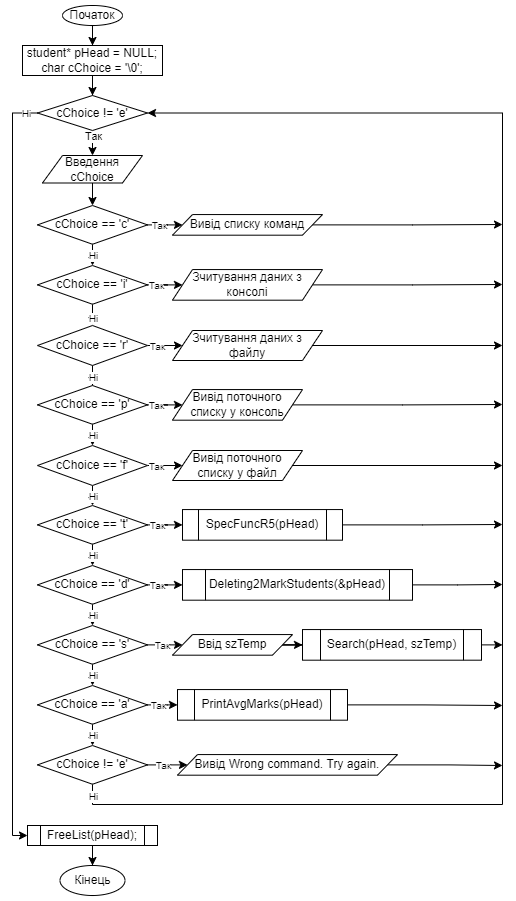


Рис.4. Функція main

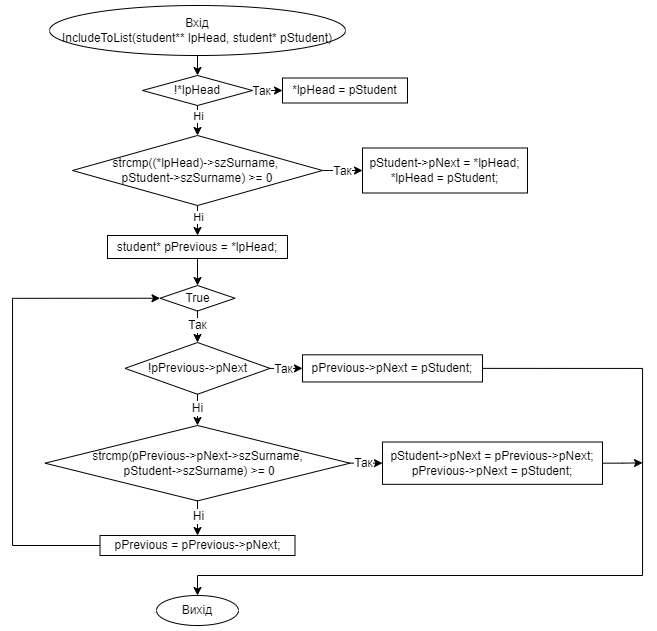


Рис.5. Функція IncludeToList

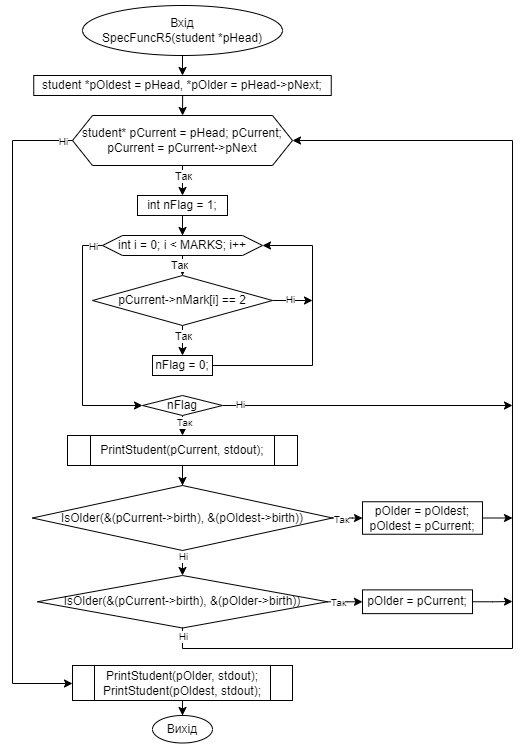


Рис.6. Функція SpecFuncR5

Текст програми:

func.h

#include <stdio.h>

#ifndef FUNC\_H

#define FUNC\_H

//number of exams

#define MARKS 8

//maximum string size

#define NAME\_SIZE 32

//contains information about the student and can make list

//uses the sDate structure for the date of birth

typedef struct sStud {

char szName[NAME\_SIZE];

char szSurname[NAME\_SIZE];

short nMark[MARKS];

struct sStud \*pNext;

struct sDate {

short nDay;

short nMonth;

short nYear;

}birth;

}student;

void FreeList(student \*pHead);

int PrintList(student \*pHead, FILE \*fStream);

int PrintStudent(student \*pStudent, FILE \*fStream);

int IncludeToList(student \*\*lpHead, student \*pStudent);

int ScanToList(student \*\*lpHead, FILE \*fStream);

int Deleting2MarkStudents(student \*\*lpHead);

int IsOlder(struct sDate\* d1, struct sDate\* d2);

int SpecFuncR5(student\* pHead);

int Search(student\* pHead, char szSegment[NAME\_SIZE]);

int PrintAvgMarks(student\* pHead);

#endif

func.c

#include "func.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

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

//frees all dynamic memory reserved by the list

void FreeList(student\* pHead) {

if (!pHead) return;

FreeList(pHead->pNext);

free(pHead);

}

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

//prints all students data that is on list to stream

int PrintList(student\* pHead, FILE \*fStream) {

if (!fStream) return 0;

if (pHead) {

PrintStudent(pHead, fStream);

PrintList(pHead->pNext, fStream);

return 1;

}else return 0;

}

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

//prints all student data to stream

int PrintStudent(student\* pStudent, FILE\* fStream) {

if (!fStream) return 0;

if (pStudent) {

fprintf(fStream, "\n %10s %10s %02hd/%02hd/%hd ",

pStudent->szName, pStudent->szSurname, pStudent->birth.nDay,

pStudent->birth.nMonth, pStudent->birth.nYear);

for (int i = 0; i < MARKS; i++) {

fprintf(fStream, "% hd", pStudent->nMark[i]);

}

return 1;

}else return 0;

}

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

//includes the student on the list with sorting by last name

//returns false if wrong pointer received

int IncludeToList(student\*\* lpHead, student\* pStudent) {

if (!lpHead || !pStudent) return 0;

if (!\*lpHead) \*lpHead = pStudent;

else if (strcmp((\*lpHead)->szSurname, pStudent->szSurname) >= 0) {

pStudent->pNext = \*lpHead;

\*lpHead = pStudent;

}else {

student\* pPrevious = \*lpHead;

while (1) {

if (!pPrevious->pNext) {

pPrevious->pNext = pStudent;

break;

}else if (strcmp(pPrevious->pNext->szSurname, pStudent->szSurname) >= 0) {

pStudent->pNext = pPrevious->pNext;

pPrevious->pNext = pStudent;

break;

}

pPrevious = pPrevious->pNext;

}

}

return 1;

}

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

//scans student from stream and include it to list

//returns false if wrong data received or a student cannot be included on the list

int ScanToList(student \*\*lpHead, FILE \*fStream) {

student\* pStudent = (student\*)malloc(sizeof(student));

if (!pStudent) {

return 0;

}

fscanf\_s(fStream, "%s %s %hd/%hd/%hd",

pStudent->szName, NAME\_SIZE,

pStudent->szSurname, NAME\_SIZE,

&pStudent->birth.nDay,

&pStudent->birth.nMonth,

&pStudent->birth.nYear);

for (int i = 0; pStudent->szName[i] != '\0'; i++) {

if (!isalpha(pStudent->szName[i])) {

free(pStudent);

return 0;

}

}

for (int i = 0; pStudent->szSurname[i] != '\0'; i++) {

if (!isalpha(pStudent->szSurname[i])) {

free(pStudent);

return 0;

}

}

if (pStudent->birth.nDay < 1 || pStudent->birth.nDay > 31 ||

pStudent->birth.nMonth < 1 || pStudent->birth.nMonth > 12 ||

pStudent->birth.nYear < 1920 || pStudent->birth.nYear > 2020) {

free(pStudent);

return 0;

}

for (int j = 0; j < MARKS; j++) {

if (!fscanf\_s(fStream, " %hd", pStudent->nMark + j) ||

pStudent->nMark[j] < 2 || pStudent->nMark[j] > 5) {

free(pStudent);

return 0;

}

}

pStudent->pNext = NULL;

if (!IncludeToList(lpHead, pStudent)) {

free(pStudent);

return 0;

}

return 1;

}

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

//removes students who received a grade of "2" in the first exam

int Deleting2MarkStudents(student \*\*lpHead) {

if (!lpHead || !\*lpHead) return 0;

for (student\* pPrevious = NULL, \*pCurrent = \*lpHead; pCurrent;) {

if (pCurrent->nMark[0] != 2) {

pPrevious = pCurrent;

pCurrent = pCurrent->pNext;

continue;

}

if (!pPrevious) {

\*lpHead = pCurrent->pNext;

free(pCurrent);

pCurrent = \*lpHead;

}else {

pPrevious->pNext = pCurrent->pNext;

free(pCurrent);

pCurrent = pPrevious->pNext;

}

}

return 1;

}

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

//compare dates, returns 1 if d1 is older, else 0

int IsOlder(struct sDate\* d1, struct sDate\* d2) {

if (d1->nYear < d2->nYear) return 1;

else if (d1->nYear > d2->nYear) return 0;

else {

if (d1->nMonth < d2->nMonth) return 1;

else if (d1->nMonth > d2->nMonth) return 0;

else {

if (d1->nDay < d2->nDay) return 1;

else return 0;

}

}

}

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

//prints students without '2' mark and looks for two oldest of them

//returns 0 if the list contains less than 2 students, otherwise 1

int SpecFuncR5(student \*pHead) {

if (!pHead || !pHead->pNext) return 0;

printf("\n Students that don't have mark '2':");

student \*pOldest = pHead, \*pOlder = pHead->pNext;

for (student\* pCurrent = pHead; pCurrent; pCurrent = pCurrent->pNext) {

int nFlag = 1;

for (int i = 0; i < MARKS; i++) {

if (pCurrent->nMark[i] == 2) {

nFlag = 0;

}

}

if (nFlag) {

PrintStudent(pCurrent, stdout);

if (IsOlder(&(pCurrent->birth), &(pOldest->birth))) {

pOlder = pOldest;

pOldest = pCurrent;

}else if (IsOlder(&(pCurrent->birth), &(pOlder->birth))) {

pOlder = pCurrent;

}

}

}

printf("\n\n The two oldest:");

PrintStudent(pOlder, stdout);

PrintStudent(pOldest, stdout);

return 1;

}

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

//searches for students by name or last name and print them

//returns 0 if not found, else 1

int Search(student\* pHead, char szSegment[NAME\_SIZE]) {

if (!pHead) return 0;

int nFlag = 0;

for (student\* pCurrent = pHead; pCurrent; pCurrent = pCurrent->pNext) {

if (strstr(pCurrent->szName, szSegment) ||

strstr(pCurrent->szSurname, szSegment)) {

PrintStudent(pCurrent, stdout);

nFlag = 1;

}

}

return nFlag;

}

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

//prints average marks from every exam

//returns 0 if wrong list received, else 1

int PrintAvgMarks(student\* pHead) {

if (!pHead) return 0;

double lfMarks[MARKS] = { 0. };

int nCounter = 0;

for (student\* pCurrent = pHead; pCurrent; pCurrent = pCurrent->pNext, nCounter++) {

for (int i = 0; i < MARKS; i++) {

lfMarks[i] += (double)pCurrent->nMark[i];

}

}

for (int i = 0; i < MARKS; i++) {

printf(" %g", lfMarks[i] / nCounter);

}

return 1;

}

Source.c

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

#include "func.h"

int main(void) {

student\* pHead = NULL;

char cChoice = '\0', garbage;

while (cChoice != 'e') {

printf("\n\n Enter the character that corresponds to the command, or c to get a list of commands: ");

while (1) {

garbage = cChoice;

cChoice = getchar();

if (cChoice == '\n') {

cChoice = garbage;

break;

}

}

switch (cChoice) {

case 'c': {

printf(" Commands list : \n\

e - exit\n\

i - add student from console\n\

r - read students from file\n\

p - print the current list\n\

f - save the current list to a file\n\

t - print students without '2' marks and two oldest of them\n\

d - delete students with the first '2' mark\n\

s - search for students by name or last name\n\

a - print the average marks from all exams\n");

break;

}

case 'i': {

if (!ScanToList(&pHead, stdin)) {

printf("\n Cannot include student. Check inputed data.");

}

while(getchar() != '\n'){}

break;

}

case 'r': {

FILE\* fp;

if (fopen\_s(&fp, "data.txt", "r")) {

printf("\n Cannot open file.");

break;

}

for (int i = 1; !feof(fp); i++) {

if (!ScanToList(&pHead, fp)) {

printf("Cannot scan %d-th line of data. Check input file.", i);

}

}

fclose(fp);

break;

}

case 'p': {

printf("\n %10s %10s %s\t %s\n",

"Name", "Surname", "Birth", "Marks");

if (!PrintList(pHead, stdout)) {

printf("\n Cannot print list.");

}

break;

}

case 'f': {

FILE\* fp;

if (fopen\_s(&fp, "out.txt", "w")) {

printf("\n Cannot open file.");

break;

}

if (!PrintList(pHead, fp)) {

printf("\n Cannot save list.");

}

fclose(fp);

break;

}

case 't': {

if (!SpecFuncR5(pHead)) {

printf("\n Cannot find oldest students.");

}

break;

}

case 'd': {

if (!Deleting2MarkStudents(&pHead)) {

printf("\n Cannot delete students.");

}

break;

}

case 's': {

char szTemp[NAME\_SIZE];

scanf\_s("%s", szTemp, NAME\_SIZE);

if (!Search(pHead, szTemp)) {

printf("\n Students not found.");

}

while (getchar() != '\n') {}

break;

}

case 'a': {

if (!PrintAvgMarks(pHead)) {

printf("\n Cannot find average marks.");

}

break;

}

default: {

if(cChoice != 'e') printf("\n Wrong command. Try again.");

break;

}

}

}

FreeList(pHead);

return 0;

}

**Висновок**

Виконуючи лабораторну роботу №2, я навчився документувати етапи проектування та кодування програми, зокрема склав схематичне зображення використаних структур даних, блок-схему алгоритмів та написав код власне самої програми.