**ЛАБОРАТОРНАЯ РАБОТА №5**

**«ПРОГРАММИРОВАНИЕ МОДУЛЕЙ С/С++»**

**Цель работы**: Изучение работы с модулями. Исследование особенностей работы с модулями С/С++.

**Вариант задания**

Описать структуру с именем STUDENT, содержащую следующие поля:

1. порядковый номер;
2. номер;
3. фамилия и имя;
4. год рождения;
5. год поступления в университет;
6. структура OCENKI, содержащая четыре поля: физика, математика, программирование, история.

Структуру данных из предыдущей лабораторной работы использовать в качестве информационного поля бинарного дерева. Написать функции:

- организации дерева,

- просмотра (обхода) дерева,

- отображения структуры дерева;

- добавления узла в дерево,

- исключения узла из дерева,

- сохранения в файл,

- загрузки данных из файла и создания по этим данным нового дерева,

- освобождения динамической памяти, занимаемой деревом (обязательно

вызывать при выходе из программы),

- выполнения действий в соответствии с вариантом, приведенным ниже.

Реализовать программу в виде нескольких модулей

**Вариант 30.** Функция, которая подсчитывает количество всех листьев дерева. Функция печати анкетных данных студентов, не получивших ни одной тройки, а если таких студентов нет, вывести соответствующее сообщение.

Значения и количество записей в таблице пользователь выбирает самостоятельно (т.е. количество строк таблицы не задается, память под узлы дерева выделяется динамически). Работу программы необходимо оформить в виде

меню.

**2. Алгоритм программы**



Рисунок 1 – Основное тело программы



Рисунок 2 - Алгоритм добавления данных в список



Рисунок 3 – Рекурсивный алгоритм добавления записи в дереве

Рисунок 4 – Функция вывода записи на экран(слева) с рекурсивным обходом (справа)



Рисунок 5 – Считывание записей с файла



Рисунок 6 ­­­­­­­­­­­­­– Поиск и редактирование записи



Рисунок 7 – Рекурсивный поиск необходимого элемента

Рисунок 8 – Рекурсивная печать данных в файл

Принцип удаления дерева из памяти работает так же как и просмотр данных в дереве за исключением того , что вместо просмотра идет удаление записи и все.

Поиск и вывод студентов работает таким же методом как и просто вывод, только существует отдельное условие, проверяющее оценки.

Поиск количества элементов и среднее арифметическое работает по принципу просмотра данных в дереве и банального возврата значения.

**3. Код программы на языке C**

**Код модуля VicMenuDLL**

#include "pch.h"

#ifndef printf

#include <stdio.h>

#endif // !printf

#ifndef strcmp

#include <string.h>

#endif // !printf

#ifndef malloc

#include <stdlib.h>

#endif // !malloc

#ifndef getch

#include <conio.h>

#endif // !getch

#define clear() system("cls");

int x, y;

/// <summary>

/// Функция получения новой позиции селектора на основе старой

/// </summary>

/// <param name="c">Код нажатого символа</param>

/// <param name="position">Указатель на массив из двух элементво (x,y)</param>

/// <param name="MaxY">Максимальная высота(глубина) списка(Макс смещение по вертикали)</param>

/// <param name="Colums">Ширина(максимальное смещение указателя по горизонтали)</param>

/// <returns>Указатель на двумерный массив, с измененными параметрами</returns>

int\* GetCurentSelector(char c // Символ клавиатуры

,int \* position // Массив в котором хранятся x и y

,int MaxY // Максимальный количество строк

, int Colums // Количество столбцов, по умолчанию - 1

)

{

int x = position[0]; int y = position[1];

switch (c)

{

case 72://вверх

if (y > 1) y--;

break;

case 80://вниз

if (y < MaxY) y++;

break;

case 75://лево

if (x > 1) x--;

break;

case 77://право

if (x < Colums) x++;

break;

}

position[0] = x; //запоминаем позицию по столбцу

position[1] = y;//позиция по строке

return position;

}

/// <summary>

/// Функция построения меню. Вызывается если нам не нужно вывести перед этим какую-то информацию.

/// </summary>

/// <param name="Menu">Указатель на массив из наименований пунктов меню</param>

/// <param name="position">Указатель на двумерный массив (x,y)</param>

/// <param name="MenuSize">Колличество элементов в массиве наименований</param>

/// <param name="Colums">Количество стобцов которое необходимо построить. Принимает значения 1,2,3</param>

/// <returns>Индекс выбранного пункта меню</returns>

int PrintMenu(char\*\* Menu, int\* position, int MenuSize, int Colums)

{

return PrintMenuWithTable (Menu, position, MenuSize, Colums, NULL, NULL, -1);

}

/// <summary>

/// Функция построения меню. Вызывается если нам нужно вывести перед этим какую-то информацию.

/// </summary>

/// <param name="Menu">Указатель на массив из наименований пунктов меню</param>

/// <param name="position">Указатель на двумерный массив (x,y)</param>

/// <param name="MenuSize">Колличество элементов в массиве наименований</param>

/// <param name="Colums">Количество стобцов которое необходимо построить. Принимает значения 1,2,3</param>

/// <param name="funcptr">Указатель на функцию вывода данных экран</param>

/// <param name="Dataptr">Указатель на информационное поле</param>

/// <param name="num">Указатель на номер элемента при выводе</param>

/// <returns>Индекс выбранного пункта меню</returns>

int PrintMenuWithTable(char\*\* Menu //Массив меню

, int\* position //Массив текущей позиции x и y

, int MenuSize //Колличество элементов меню

,int Colums,//Количество столбцов

int (\* funcptr ) (void\*, int),

void \* Dataptr,

int num

)

{

int Rows = MenuSize; int Override = 0; int Console\_Width = 180;

if (Colums > 3) Colums = 3; //Если ввели неверное значение строк (больше 3-х) то ставим 3

if (Colums < 1) Colums = 1;// если строк меньше одной - ставим одну

if (Colums > 1) { Rows = MenuSize / Colums; Override = MenuSize % Colums; if (Override > 0) Rows++; } //Если строк больше одной , вычисляем количество не полных строк

while (1) //цикл отрисовки меню

{

int flag = 0; //флаг является показателем того что вывод строки окончен

if (funcptr != NULL && Dataptr != NULL) //Если у нас есть данные для вывода - выводим

funcptr (Dataptr, num);

for (int i = 0; i < Console\_Width; i++) //цикл вывода делителя

{

printf("-");

}

printf("\n"); //переход на следующую строку

if (Colums == 1) //если столбцов всего 1

{

for (int i = 0; i < MenuSize; i++) // до конца массива меню

{

if (position[1] - 1 == i) // если курсор на позици

printf("| --> %-172s |\n", Menu[i]);//выводим с курсором

else printf("| %-176s |\n" ,Menu[i]);// иначе без

}

} else

if (Colums == 2) // если столбцов 2 , то

{

int PosOut = 0; //Переменная текущей позиции в массиве меню

for (int i = 0; i < Rows; i++) { //цикл прохода по строкам

for (int j = 0; j < Colums; j++) // цикл по столбцам

{

printf("|"); // делитель

if (i == Rows - 1 ) // если это последняя строка

{

if (Override) { // если она не полная , то

if ((position[0] - 1 == j)&&(position[1] - 1 ==i)) printf(" --> %-83s", Menu[PosOut++]); //если курсор на текукщей позици, выведем с ним

else printf(" %-86s ", Menu[PosOut++]); // иначе без

printf("|"); for (int l = 1; l < 89; l++) printf(" "); printf("|"); // печатаем делители

flag = 1; // вывод строки меню завершен

break; // остановить цикл

} else { //если все строки полные

if ((position[0] - 1 == j) && (position[1] - 1 == i)) printf(" --> %-83s", Menu[PosOut++]); // если есть селектор - выводим с ним

else printf(" %-86s ", Menu[PosOut++]); // иначе без

}

}

else // если строка не последняя

{

if ((position[0]- 1 == j) && (position[1] - 1 == i)) printf(" --> %-83s", Menu[PosOut++]); // если селектор стоит на текущей позиции выводим с ним

else printf(" %-86s ", Menu[PosOut++]); // иначе без

}

}

if (!flag) // если достигли конца , делаем просто переход без делителя

printf("|\n"); else printf("\n");

}

}

else // если строк больше 2

{

int PosOut = 0;

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

for (int j = 0; j < Colums; j++)

{

printf("|");

if (i == Rows - 1)

{

if (Override == 1) {

if ((position[0] - 1 == j) && (position[1] - 1 == i)) printf(" --> %-54s", Menu[PosOut++]);

else printf(" %-57s ", Menu[PosOut++]);

printf("|"); for (int l = 1; l < 60; l++) printf(" "); printf("|");

for (int l = 1; l < 60; l++) printf(" "); printf("|");

flag = 1;

break;

}

else if (Override) {

if ((position[0] - 1 == j) && (position[1] - 1 == i)) printf(" --> %-54s", Menu[PosOut++]);

else printf(" %-57s ", Menu[PosOut++]); printf("|"); j++; i++;

if ((position[0] - 1 == j) && (position[1] - 1 == i)) printf(" --> %-54s", Menu[PosOut++]);

else printf(" %-57s ", Menu[PosOut++]);

printf("|");

for (int l = 1; l < 60; l++) printf(" "); printf("|");

flag = 1;

break;

}

else {

if ((position[0] - 1 == j) && (position[1] - 1 == i)) printf(" --> %-54s", Menu[PosOut++]);

else printf(" %-57s ", Menu[PosOut++]);

}

}

else

{

if ((position[0] - 1 == j) && (position[1] - 1 == i)) printf(" --> %-54s", Menu[PosOut++]);

else printf(" %-57s ", Menu[PosOut++]);

}

}

if (!flag)

printf("|\n"); else printf("\n");

}

}

for (int i = 0; i < Console\_Width; i++) {

printf("-");

}

char c = getch();

if (c == 13) {

if (Colums == 1) return position[1]; else if (Colums == 2)

return position[0] + (position[1] - 1) \* 2; else return position[0] + (position[1] - 1) \* 3;

}

int MaxY = MenuSize / Colums; if (Override) MaxY++;

position = GetCurentSelector(c, position, MaxY, Colums);

clear();

}

}

const char\* BMP1[] = {

"................~J?!~~~~!!7?JYYYY55555PPPPPPPPPPPPPPPPPPGPPPPPPGBBBBBBBB#&&#B##BGB#BGB#BGGGGGGBGGGGGGGGGGGGGG5^75GGGB#&B7:^!7!777JGPB######B7......................................." ,

"..................!JJ7~^^^:::^^~^::^~^^^~~~!!~!7?JYY55PGPPPPPGGGGPPPPG##&&#BGGBGGGBGGGGGGGGGGGGGGGGGGGGGGGGGGY. .^?PGB#&#GJ!!!777PGG#######BG7......................................" ,

" ...........^?J?!~~~^:::...... . .:^~JGPPPPPPGPPPPPPGB###BGPPPPPPPPPPGGGGPJ?5GGGGGGGGGGGGGG5!?J!.~55GB###&#P7!775GPG#B######BP7:................................. " ,

" .... .:?Y7!~~~^:....::.. ~J5PPPP55555555GBB##BPPPPPPPPPPPPPPGGG7^:!YPGGGGGGGGGGY^~!~JPPGGPB#####&BJ75GPPB&##B###BGG57........................ .." ,

".......... ...:7YJ?77~^:....... .:7P5P555555555PBPPGBP55PPPPPPPPPPPPPPPJ.. .^57JGGYYY!:...:7PGPPPPG#####&&#BGPPG&####GB##GGG57............... .........." ,

".... ...........:!Y5Y7~^^:::::. .:^Y??5555555555BP!PGG5555PPPPPPPPPPPPPGJ. .7.:~YP???::^!5GPPPPPPG#####&#BGGGB##B##BGGB#GGGY!..... .................." ,

". ...... ..............~JY?~::::..... .:^P7^Y55555555YGP.~5BP55555PPPPPPPPPPPGGG! .: ~PJYGPGGGGGGGGGGGB####B#BGGGBGGGGGBBGGB#GGPJ~ ..........................." ,

". . .................^JP5Y!:::. ::PJ~5YY555555Y5G: 7YP?75555PPPPPPPPP5Y!~5G^ .YGGGGGGGGGGGBBBB#####BB##BGGGGGGGGGGGGBGG5?^................................" ,

".... ......................^?5PJ!:.. .:5P?5Y!JJ~~YYY!5~ 77J7.:::55?J7J?5PPYY?JPGG:..........:?GGGGGGGGGGGBGGGGGGGGGGB##BBBBGGGGGGGGGGGJ?................................" ,

"....................................~JP5!:......^YG5P555Y~.:^^!!7 .7!??. .:YP!7?55PGGGGBGGB#P..........~7JGBBBBBBBGGGGGGGGGGGGGGGBBBBBBBBBBBBBBGBP?!..............................." ,

".................................... .:!YP5J?~:7YGPBG5555JYYYY^~. .7~?Y7Y55PPPGGGGGGGGGBBB##&Y.::.:::^^.!??G#BGGGBGGGGGGGGGGGGGGGGBBGGGGGGGGBBGGBGY?~.............................." ,

"....................................... .:^7JJJPPPGBBP5555555Y.. .7~75BBPPPPPPGGGGGGGGB####&G!^^^~!^^~::~^!PBGGGGGGGGGGGGGGGGGGGGGB#BGGGGGGGGGGGGPJ?.............................." ,

".......................................... ..~GGGBBBG5555555? .7~?5BBBBBGGGGGGBBBBB##BB#&5~~^~7~:::::::^~YBBBGGGGGGGGGGGGGGGGGGGB#BBBGGGBBBBBBGP57............................." ,

"..............................................YBBB##BGPPPPPPP!:. ...!!75B#####BBB##BBBBB##BBB#Y:.:^!?~^^^^:::~^?BBBBBBBBGGGGGGGGGGGGGGBBB#BBBBBBBBBBGP5^............................" ,

"............................................ ~##GGGGBGGGGGGGP77::..:~7^7B#####BBB##BBBBB##G#B#?: :~!:.:^^^^^^7:!G###BBBBBBBGBBBBBBBBBBBBBB###BBBBBBBGPY............................" ,

"........................................... .B&BPPPPPGGGGGGGG!7.:::^~!::PBBBGBBBBBBBBBBB##5###! :7! .~!~~~JJ~^JG###BBBB#BBBBBBBBBBBBBB#BBB####BBBBPG! .........................." ,

"...........................................:Y#&BPGGPGGGGGGGGP:!..:.!~~: 7#BBBBBBBBBBBBBB#BYB#G^ :?. :!!~!~7!^~YB####BBB####BBBBBBBBBB###BBB###BBB55..........................." ,

".......................................:::!GG##GGGGGGGGGGGGGP:7::^^7~^. P#BBBBBBBBBBBBBBPJG#J. ^: ^!!:.:~!~!YG##&&#########BBBBBBBB####BB##B#G57.........................." ,

".................................:..:::::!PPG&BPGBGGGGGGGGBBG^?^^!~^~^ .G#BBBBBBBBBGGGGY?PB. . :?. .^~?7?5B&&@&&&&&&&&&&#####B##########55:........................." ,

"....................... ....... .^^:!5PPPB#GGGBGGGGBBBBBBG^?~~?!.:: ^BBBBBGGGGGGPPP??GJ ..:~7J5PPBG5Y?^.:5&GJ?YG#&&&&&&&&&&&&&&&&&&##&&&###B5J........................." ,

"...................... .... .::^?PPPPPBBGGBBGBBBBBBB#B5:JJ7Y!.:^ ^PBGGGGPPGGPPY?75~ .~?Y5B&&&&&&&&@@@@&7:#@#BG5YPB&&&####&&&&&&&&&&&&&&####P5!........................" ,

".................... ... ^:^77^^~!~!5GGB#BBBBBBBBB&#Y^?7J5~ . :!PGPPP55P55J?^!^ .7BP7:.P&&#BGBGGB#B#@: 7@&####BBBB######BBB#############BPPP^ ......................" ,

"..................... ::....:!Y?~. .^^!5GG#&&BBBBBBBB#&#J~!^^J~ .JP555YYYY?7.:: :J!. :7YJJY#J7!~!G! .&&##############################G5GPG: ....................." ,

"..................... . .^!?YY~. :^7YPPG##&#BBB##BB#&&B7~~: : ~??JJ?7?7: .. . Y~:^:....^~: .#&##############################BBBBGG: ...................." ,

"...................... . .. .:~YGGGB#B#&BBBB####&&&G~!^ :7Y55GGBGP5!. :~777!!. :?:. ... ~#&######&#&&############GB##########BGG: ..................." ,

"...................... . . .:^^!Y5PGGPG##BBB####&&&#Y~! 7B#GJ7##BBBGBBY. :~77~ .:. Y########&#&&###########BGG###########BBB~ .................." ,

"....................... . .:^~^JBBB#&#&&@@&B!7#!~#@7 ^JB57YY!?~ .:^ !G&####&##&#B&&############BB###########B#B7.................." ,

"....................... ...:^^^^GBB#&&&&@@&#5~B@@5.:~:. 7P..:: .~ !P&B###&##&&#B&&############################BJ................." ,

"....:.................. .?PPG5^YBB#&&&@@@@&B!Y@&&&P .!:. ... :!75G&&&&#B#&#B#&&&#B############&########BBB#&#5^..............." ,

"....:................... :B##GJBB#&&@@@&&&&Y7&&&#&#G: .. ...:YPB###BB#&##B&&&&&BBB##B#######&&##BB##BBBBB#&#G?:............." ,

".....:................... J###BB&&&@@@&&&&BJ&@&&#BBBG: ^?JY5GBB#&&&#B#&#&&&#BB##BBBB##B#&&&##BB##BBBB###&#P7:. ........." ,

".....:^::......::::^~~7?5~ .P&#&&&&&&@&&&&#G&@&&&#BBBGP: .~!7?5G##&&&&#B#&&#&&####&&##BB##B&&&&&&##B###BBB#BGB##GJ~........." ,

".......:^^~?YYYYYY55PGGGGG~ ~B&&&&&@@&&&&&&&@@&&##BBBGGP~. .:^~!?YGG#&&&&BPB#&&BB&&B#&#&&##BB###&&&&&&&##BB#######GPGBGP5?~:....." ,

".........::~?Y555PPGGGBBBBB. :~B&&@@@&&&&@@@&&@&&&#BBBGGGG?~:. .....^JJG&&&B55GG55PGB&&B#BB#&&&#B###&#&&&&&&###BB####&##G5J7!7???7!~^" ,

"..............^~?YPGBBGGGGG? .^?&@@&&&&&@@&&&&&@@&&#BBBGG5?^ :7JB&#G5Y5YJ?????JYPG#BB####&&###BB#&&&&&5?GBBBB####&&#P5J7~^:...." ,

"...........:~?YPGGP555YY55Y5^. .^?&@@@@&&&&&&&&&&&&&&&#BBGP5YJ!^. .!5P##PYJJ??7777777!!?G#BB#BGB&&##BBG&&&&&&&^.!YPPGB##B##&#BPYJ?7!^:" ,

".........^?5PGGPYJJJ?~..:!JJ~^^:. .::^!B@&@&#B&&&&&&&&&&&&&&&&#BBP5Y?77!!~^:. :^~5GG5J???777!~~~!7!~^7PBGGBGYG##BBBBB&&&&&&&#. .:~!?J55PPPGGGG5YJJJJ" ,

"......:^^JGP5YJ??JJ~ ..:^~~~^^^~~~!!7B&@&GG&@&&&&&&&&&#&&&&&&&&##BGG5JY!: .:^^^~!~7J7!!~^^^~~!7?7!!7?GGGGBPJGBGGBBB#&&##&&&&B: ......::^~~~!!7777!!" ,

".....^^~!YJ7!7??J?. .:^~!!~7!.Y#@#PP&@&&&@#&@&&#G#@@&BBGPPGGGP5YYYJ?!:. ..:^::::::~?!^::^~~~^^^^~7??JJGGGGB?YBGGGBBB&&&#BB#&&@#!. ...... ........." ,

"...:^~7?7!!!!!77~. ....:^!~.~G&#55#@&&&@&B&@&#GPPBGY?7!!~~~!!!77??????7!~:. ..:::::...:^~!:. ... ......:~?5GGGB5!PGGGGBGB&&&&BBB#&&&&G^. ....^J^ ......" ,

"..:^!?7!!!!!!!!. ..:::::::^YG#55P&&&&@&B#@&BBPJ??7!~^^::::::::^~~^^~~!!777!!^:. ..::::::.....:^^:. ~YGGGGP7?GGGGBBB&&&&&B#####&&&&GY?7?JY5PJ:::~!..." ,

".:~J?!!!!!!~~^. ...::^^^^^?PP55#@&@@@&G#@&GGGPJ7~^^^^^~~^:. ..^^^^^^^^~~~~!!^^:::::::^^.. .:. ^YPGGG!!?5BGGGBB&@&&&#BB##&&&&&&&&&&#BGPY7777^...." ,

":!J7!!!!~^:.. ...::^^::PP555&@@@&@&GB@&PPG?!~~~!!7~:. .::^^:::::....:^:. :: .. :J5GGY~:7JGGGGBB?7B@@&#BBB#&&&&&&&&&&#BBGY7~:......" ,

"!J!!!~:.... ...::^^.:5P55P&&@&&@&GG&&PPJ!!!!!7~. . . ~YPPY!^^~J5GGBBP!::~&@@&&&#####&&&&&&&&&&&&G?^. ...." ,

"?!!^...... ...::^^.:!P555&@&B&@@#PB@#Y~~~~!7^ ^?J?^..:^^?5BBGY7~~?P&@@&&&&&######&&&&&&&&&&@@&#J:..." ,

"7!:........ ...::^::..YP55&@GB@&@&BP#B^:^~!7~. . .:~!~:....:^:!55YJ?!~^^~5&@@&&&&&&&&&&&&&&&&&&&&&&&&&@&5:." ,

"7~.........~7: ...::^:: .YGP5G&5G&&&@@BY~^!!!7!^..:. . . ..:::^^^!!~~~77?G##&&&&&@@@@@@&&&&&&&&&&&&&&&&&&&&#P!" ,

"J^........^5#BP!. ...::::.::?PPPPPB5P&@@&@G7!?7^^!^:::. .. .:.. .... .:^!!!!~~~~~~~~^~!J#&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&BGG5" ,

"^:.......^5BGGGGPY!: ..::::.7JY5YJY5PGP5B@&&@@&J^:^7^::^: ... .: . .:^~~^^~^^:::::^^~77?7!~^^^^^^^^^!#&&&&&&&&&&&&&&&&&&&&&&&&#B&&&#P5PP"

, NULL};

/// <summary>

/// Анимированная картинка в консоли

/// </summary>

void animatedNeko() {

FILE\* f; int count = 1;

while (1) {

for (int i = 0; i < 11; i++) {

char url[256] = { 0 };

sprintf(url, "bakemonogatari-monogatari/banner (%d).txt", i);

f = fopen(url, "r"); char a[195]; ;

while (fgets(a,195,f) != NULL)

{

printf("%s", a);

}

Sleep(30);

COORD positionCur = { 0,0 }; //позиция x и y

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleCursorPosition(hConsole, positionCur);

}

count++;

if (count >= 20) return;

}

}

/// <summary>

/// Картинка в консоли

/// </summary>

/// <param name="Password">Пароль для вывода</param>

void neko(int Password) {

if (Password == 34097) {

clear();

for (int i = 0; BMP1[i] != NULL; i++) {

printf("%s\n", BMP1[i]);

}

}

}

**Код модуля TreeStruct**

#pragma once

typedef struct {

int id;

char name[40];

char surname[40];

int birth\_date;

int stud\_date;

struct {

int fiz;

int math;

int it;

int history;

} marks;

} STudent;

/// <summary>

/// Структура ноды дерева

/// </summary>

typedef struct Persons {

STudent info;

struct Persons\* right;

struct Persons\* left;

} Person;

**Основной код**

#pragma warning(disable : 4996);

#include "E:\Учеба универ\А и П\Biblis\VicMenuDLL\VicMenuDLL\VicMenuDLL.h"; //файл библиотеки меню

#ifndef printf

#include <stdio.h>

#endif // !printf

#ifndef strcmp

#include <string.h>

#endif // !printf

#ifndef malloc

#include <stdlib.h>

#endif // !malloc

#ifndef getch

#include <conio.h>

#endif // !malloc

#include "ThreeStruct.h"

#define clear() system("cls");

const char\* Menu[] = { "Добавить новый элемент",

"Загрузить из файла",

"Записать все в файл ",

"Вывод данных в списке",

"Редактировать запись",

"Отобразить структуру дерева",

"Удалить элемент(любой)",

"Анимация",

"Картинка",

"Очистить дерево",

"Колличество элементов в дереве",

"Отображение всех записей в которых нет ниодной тройки",

"Среднее арифметическое по всем предметам",

"Выход" };

const int SizeStudent = sizeof(STudent);

const int MenuSize = 14;

HANDLE hConsole;

//------------------------------------------------------------------- Область функций ----------------------------------------------------------------

//-----------------------------------функции создания дерева----------------------------------------------

/// <summary>

/// Функция добавления элемента в дерево

/// </summary>

/// <param name="head">Указатель на корень лерева</param>

/// <param name="info">Информация которую необходимо сохранить</param>

void addToTree(Person\*\* head, const STudent\* info);

/// <summary>

/// Функция вывода меню добавления записи

/// </summary>

/// <param name="st">Указатель на корень дерева</param>

/// <returns>При успешном добавлении возвращает 1</returns>

int аddNewElement(Person\*\* st);

/// <summary>

/// Функция реализующая ввод через форму

/// </summary>

/// <param name="d">Информационные поля</param>

/// <returns>Возвращает информационную структуру которую необходимо сохранить</returns>

STudent GetInfoFromCeyboard(STudent d);

/// <summary>

/// Отображение структуры дерева

/// </summary>

/// <param name="top">Параметр корня дерева/текущей ноды</param>

/// <param name="otstup">Текущий отступ вывода</param>

void otobr(Person\* top, int otstup);

//----------------------------------Функции работы с файлами----------------------------------------------

/// <summary>

/// Функция сохранения дерева в файл

/// </summary>

/// <param name="f">Указатель на файл</param>

/// <param name="St">Корень дерева</param>

/// <returns>В случае успеха вернет 1</returns>

int create\_file(FILE\* f, Person\* St);

/// <summary>

/// Функция загрузки данных из файла в дерево

/// </summary>

/// <param name="f">Указатель на файл</param>

/// <returns>Возвращает указатель на корень дерева</returns>

Person\* loadFromFile\_new(FILE\* f);

/// <summary>

/// Рекурсивная функция сохранения данных в файл

/// </summary>

/// <param name="f">Указатель на файл</param>

/// <param name="St">Корень дерева/текущая нода</param>

void printToFile(FILE\* f, Person\* St);

//------------------------------Функции работы с элементами дерева----------------------------------------

/// <summary>

/// Основное тело функции редактирования записи по ид

/// </summary>

/// <param name="st">Указатель на корень дерева</param>

/// <returns>В случае успеха - 1</returns>

int correctInfo(Person\* st);

/// <summary>

/// Вспомогательная рекурсивная функция поиска необходимого листка/ноды

/// </summary>

/// <param name="root">Корень дерева или текущаяя нода</param>

/// <param name="indexToSerch">Ид который необходимо найти</param>

/// <returns>Возвращает указатель на элемент с соответствующим ид, иначе NULL</returns>

Person\* getLeaf(Person\* root, int indexToSerch);//поиск листка по ид

/// <summary>

/// Рекурсивная функция удаления дерева из памяти

/// </summary>

/// <param name="st">Корень дерева/текущаяя нода</param>

/// <returns>Пустой указатель на структуру</returns>

Person\* deleteThree(Person\* st);

/// <summary>

/// Рекурсивная функция поиска и удаления ноды/листа из дерева по ид

/// </summary>

/// <param name="root">Корень дерева/текущая нода</param>

/// <param name="id">Значение которое ищем в дереве</param>

/// <returns>Корень дерева после изменения</returns>

Person\* DeleteNode(Person\* root, int id);

/// <summary>

/// Рекурсивная функция подсчета количества записей в дереве

/// </summary>

/// <param name="root">Корень дерева/текущаяя нода</param>

/// <param name="count">Используется для рекурсивной передачи данных</param>

/// <returns>Количество элементво в дереве</returns>

int getLeafCount(Person\* root, int count);

/// <summary>

/// Рекурсивная функция подсчета среднего арефметического по предметам

/// </summary>

/// <param name="root">Корень дерева/текущая нода</param>

/// <param name="summ">Указатель на массив флоат в котором хранятся значения</param>

/// <returns>Указатель на массив из четырех элементов типа флоат где лежит сумма балов по предметам</returns>

float \* getSerArefm(Person\* root, float \* summ);

//--------------------------------Функции вывода списка на экран------------------------------------------

/// <summary>

/// Рекурсивная функция печати информации из дерева на экран. Обход с лева на право.

/// </summary>

/// <param name="st">Корень дерева/текущая нода</param>

/// <returns>В случае успеха - 1</returns>

int PrintTreeData(Person\* st);

/// <summary>

/// Форма вывода записи на экран

/// </summary>

/// <param name="d">Указатель на сируктуру с информационными полями</param>

/// <param name="index">Текущий номер элемента</param>

int PrintTreeDataNonThree(Person\* root);

//-------------------------------Вспомогательные функции -------------------------------------------------

/// <summary>

/// Функция выбора необходимой операции. Использует для получения вызов библиотеки VicMenuDLL

/// </summary>

/// <param name="selector">Индекс выбранного элемента</param>

/// <param name="f">Указатель на файл</param>

void MenuSelect(int selector, FILE\* f);

/// <summary>

/// Функция получения подтверждения от пользователя

/// </summary>

/// <returns>Если подвердили - вернет 1</returns>

int getResponse();

/// <summary>

/// Функция реализующая ввод через форму

/// </summary>

/// <param name="d">Информационные поля</param>

/// <returns>Возвращает информационную структуру которую необходимо сохранить</returns>

STudent GetInfoFromCeyboard(STudent d);

/// <summary>

/// Заголовок таблицы

/// </summary>

int printTable();

/// <summary>

/// Горизонтальный разделитель таблицы

/// </summary>

int printBorder();

int correctInfo(Person\* st);

/// <summary>

/// Вспомогательная рекурсивная функция поиска необходимого листка/ноды

/// </summary>

/// <param name="root">Корень дерева или текущаяя нода</param>

/// <param name="indexToSerch">Ид который необходимо найти</param>

/// <returns>Возвращает указатель на элемент с соответствующим ид, иначе NULL</returns>

Person\* getLeaf(Person\* root, int indexToSerch);//поиск листка по ид

/// <summary>

/// Рекурсивная функция удаления дерева из памяти

/// </summary>

/// <param name="st">Корень дерева/текущаяя нода</param>

/// <returns>Пустой указатель на структуру</returns>

Person\* deleteThree(Person\* st);

/// <summary>

/// Рекурсивная функция поиска и удаления ноды/листа из дерева по ид

/// </summary>

/// <param name="root">Корень дерева/текущая нода</param>

/// <param name="id">Значение которое ищем в дереве</param>

/// <returns>Корень дерева после изменения</returns>

Person\* DeleteNode(Person\* root, int id);

/// <summary>

/// Рекурсивная функция подсчета количества записей в дереве

/// </summary>

/// <param name="root">Корень дерева/текущаяя нода</param>

/// <param name="count">Используется для рекурсивной передачи данных</param>

/// <returns>Количество элементво в дереве</returns>

int getLeafCount(Person\* root, int count);

/// <summary>

/// Рекурсивная функция подсчета среднего арефметического по предметам

/// </summary>

/// <param name="root">Корень дерева/текущая нода</param>

/// <param name="summ">Указатель на массив флоат в котором хранятся значения</param>

/// <returns>Указатель на массив из четырех элементов типа флоат где лежит сумма балов по предметам</returns>

float\* getSerArefm(Person\* root, float\* summ);

void printINFO(STudent \*d, int index);

/// <summary>

/// Рекурсивная функция вывода информации о студентах без троек

/// </summary>

/// <param name="root">Корень дерева/текущая нода</param>

/// <returns>В случае успеха - значение > 1 </returns>

//------------------------------------------------------------------- Конец области функций ---------------------------------------------------------

Person\* st = NULL;

Person\* root = NULL;

int position[] = { 1,1 };

int main(void) {

hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleCP(1251); // Задаем таблицу символов для консоли.

SetConsoleOutputCP(1251);

// system("color F0");

FILE\* f = fopen("data.dat", "rb+");//Открытие существующего файла для чтения и записи в конец

if (!f) {

f = fopen("data.dat", "wb+"); //Создание нового файла для обновления

if (!f) {

puts("Не могу открыть (создать) файл\n");

return 1;

}

}

while (1) {//вывод меню и запуск соответствующих функций

clear();

MenuSelect(PrintMenu(Menu, position, MenuSize, 2), f);

}

}

int Posid = 1;

void MenuSelect(int selector, FILE\* f)

{

switch (selector) {

case 1:

аddNewElement(&st);

break;

case 2:

if (getResponse()) {

st = loadFromFile\_new(f);

}

break;

case 3:

create\_file(f, st);

break;

case 4:

clear();

Posid = 1;

PrintTreeData(st);

puts("Нажмите любую клавишу");

getch();

break;

case 5:

correctInfo(st);

break;

case 6:

clear();

printf("---------------------------------------------- Структура дерева -----------------------------------------\n");

otobr(st, 1);

printf("-------------------------------------------- Конец струк. дерева ----------------------------------------\n");

getch();

break;

case 7:

if (getResponse()) {

printf("\nВведите ИД записи -->"); int l; scanf("%d", &l);

st = DeleteNode(st, l);

puts("Ветка удалена ... Нажмите любую кнопку..."); getch();

}

clear();

break;

case 8:

if (getResponse()) {

animatedNeko();

}

break;

case 9:

if (getResponse()) {

neko(34097);

getch();

}

break;

case 10:

if (getResponse()) {

st = deleteThree(st);

printf("\n\n Дерево очищенно, нажмите любую кнопку...");

getch();

}

break;

case 11:

if (getResponse()) {

printf("\n\n Дерево содержит %d записей. Нажмите любую кнопку...",getLeafCount(st,0));

getch();

}

break;

case 12:

Posid = 1; printf("\n");

if (PrintTreeDataNonThree(st)== 1)

printf("\n\n Студентов без троек нет. Нажмите любую кнопку...");

getch();

break;

case 13:

if (getResponse()) {

float summ[] = { 0,0,0,0 };

getSerArefm(st, &summ);

int l = getLeafCount(st, 0);

float temp = summ[0] / l;

printf("\n Средняя оценка по Физике --> %.4f ", temp);

temp = summ[1] / l;

printf("\n Средняя оценка по Математика --> %.4f ", temp);

temp = summ[2] / l;

printf("\n Средняя оценка по Информатика --> %.4f ", temp);

temp = summ[3] / l;

printf("\n Средняя оценка по История --> %.4f ", temp);

getch();

}

break;

case 14:

if (getResponse()) {

if (!st) {

deleteThree(st);

}

exit(666);

}

break;

}

}

int create\_file(FILE\* f, Person\* root)

{

if (!root) return 666;

fclose(f); f = fopen("data.dat", "wb+");

fseek(f, 0, SEEK\_SET);

printToFile(f, root);

clear();

puts("Данные сохранены. Нажмите любую кнопку ....");

getch();

return 0;

}

void printToFile(FILE\* f, Person\* root)

{

if (root) {

STudent te = root->info;

fwrite(&te, SizeStudent, 1, f);

if (root->left) {

printToFile(f,root->left);

}

if (root->right) {

printToFile(f,root->right);

}

}

}

int PrintTreeData(Person\* root) {

if (root) {

if (root->left) {

PrintTreeData(root->left);

}

printINFO(&root->info, Posid);

Posid++;

if (root->right) {

PrintTreeData(root->right);

}

}

return 0;

}

Person \* deleteThree(Person\* root) {

if (root) {

if (root->left) {

deleteThree(root->left);

}

if (root->right) {

deleteThree(root->right);

}

free(root);

}

return NULL;

}

void addToTree(Person\*\* root, const STudent\* info)

{

if (\*root == NULL)

{

\*root = (Person\*)calloc(sizeof(Person), 1);

(\*root)->info = \*info;

}

else {

if ((\*root)->info.id > info->id)

addToTree(&((\*root)->left), info);

else

addToTree(&((\*root)->right), info);

}

}

int аddNewElement(Person\*\* st)

{

STudent d; d.id = NULL; d.birth\_date = NULL;

char\*\* Menu[] = { "Добавить запись","Выход" };

int position[] = { 1,1 }; int flag = 0;

while (1) {

clear();

if (flag)

{

switch (PrintMenuWithTable(Menu, position, 2, 1, printINFO, &d, 1)) {

case 1:

{

d.id = NULL; d.birth\_date = NULL;

d = GetInfoFromCeyboard(d);

addToTree(st, &d); flag = 1;

}

break;

case 2:

return 1;

}

} else

switch (PrintMenu(Menu, position, 2, 1)) {

case 1:

{

d = GetInfoFromCeyboard(d);

addToTree(st, &d); flag = 1;

}

break;

case 2:

return 1;

}

}

}

void otobr(Person\* top, int otstup) {

if (top) {

otstup += 3; //отступ от края экрана

otobr(top->right, otstup); //обход правого поддерева

for (int i = 0; i < otstup; i++) printf(" ");

printf("|%d\n", top->info.id);

otobr(top->left, otstup); //обход левого поддерева

}

}

Person\* loadFromFile\_new(FILE\* f)

{

STudent tmp;

Person\* head = NULL;

int count = 1;

fseek(f, 0, SEEK\_SET);

printf("\n"); // printTable();

while (fread(&tmp, sizeof(STudent), 1, f))

{

addToTree(&head, &tmp);

printINFO(&tmp,count);

count++;

}

puts("Данные считаны. Нажмите любую кнопку ");

getch();

return head;

}

Person\* DeleteNode(Person\* root, int id) {

if (root == NULL) return root; // выход если пустой узел

if (root->info.id == id) { //найден удал. узел

Person\* tmp; // указатель

if (root->right == NULL) tmp = root->left;

else { // существует правое поддерево

Person\* ptr = root->right;

if (ptr->left == NULL) { // у правого ПД отсутствует левое ПД

ptr->left = root->left;

tmp = ptr;

}

else {

Person\* pmin = ptr->left; // поиск самого левого

while (pmin->left != NULL) {// узла в правом ПД

ptr = pmin;

pmin = ptr->left;

} // найден самый левый узел правого ПП (pmin)

ptr->left = pmin->right;

pmin->left = root->left;

pmin->right = root->right;

tmp = pmin;

}

}

free(root);

return tmp;

}

else //бинарный поиск в левом или правом поддереве

if (id < root->info.id)

root->left = DeleteNode(root->left, id);

else

root->right = DeleteNode(root->right, id);

return root;

}

int getLeafCount(Person\* root, int count) {

if (root) {

if (root->left) {

count = getLeafCount(root->left,count);

}

count++;

if (root->right) {

count = getLeafCount(root->right, count);

}

}

return count;

}

int PrintTreeDataNonThree(Person\* root) {

if (root) {

if (root->left) {

PrintTreeDataNonThree(root->left);

}

if ((root->info.marks.fiz > 3) && (root->info.marks.math > 3) && (root->info.marks.it > 3) && (root->info.marks.history > 3))

{

printINFO(&root->info, Posid);

Posid++;

}

if (root->right) {

PrintTreeDataNonThree(root->right);

}

}

return Posid;

}

float \* getSerArefm(Person\* root, float \* summ) {

if (root) {

if (root->left) {

summ = getSerArefm(root->left, summ);

}

summ[0] = summ[0] + root->info.marks.fiz;

summ[1] = summ[1] + root->info.marks.math;

summ[2] = summ[2] + root->info.marks.it;

summ[3] = summ[3] + root->info.marks.history;

if (root->right) {

summ = getSerArefm(root->right, summ);

}

}

return summ;

}

int getResponse() {

printf("\n Вы уверены что хотите выполнить данную команду ? [Y | Any] "); char c = getch();

if ((c == 'y') || (c == 'Y')) return 1; else return 0;

}

STudent GetInfoFromCeyboard(STudent d)

{

hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

char del = '|'; int index = 1;

if ((d.id != NULL) || (d.birth\_date != NULL))

{

int\* position = (int\*)malloc(2 \* sizeof(int));

position[0] = 1; position[1] = 1; int tempPosition = 1; int flag = 0;

char c;

while (1) {

clear(); printTable();

switch (position[0]) {

case 1:

case 2: {

printf("%c %-3d %c->%-5d %c %-32s %c %-30s %c %-14d %c %-17d %c %-8d %c %-12d %c %-18d %c %-9d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\nВыход из редактирования\n");

break;

}

case 3: {

printf("%c %-3d %c %-6d %c->%-31s %c %-30s %c %-14d %c %-17d %c %-8d %c %-12d %c %-18d %c %-9d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\nВыход из редактирования\n");

break;

}

case 4: {

printf("%c %-3d %c %-6d %c %-32s %c->%-29s %c %-14d %c %-17d %c %-8d %c %-12d %c %-18d %c %-9d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\nВыход из редактирования\n");

break;

}

case 5: {

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c->%-13d %c %-17d %c %-8d %c %-12d %c %-18d %c %-9d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\nВыход из редактирования\n");

break;

}

case 6: {

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c->%-16d %c %-8d %c %-12d %c %-18d %c %-9d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\nВыход из редактирования\n");

break;

}

case 7: {

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c->%-7d %c %-12d %c %-18d %c %-9d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\nВыход из редактирования\n");

break;

}

case 8: {

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c %-8d %c->%-11d %c %-18d %c %-9d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\nВыход из редактирования\n");

break;

}

case 9: {

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c %-8d %c %-12d %c->%-17d %c %-9d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\nВыход из редактирования\n");

break;

}

case 10: {

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c %-8d %c %-12d %c %-18d %c->%-8d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\nВыход из редактирования\n");

break;

}

case 11: {

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c %-8d %c %-12d %c %-18d %c %-9d %c\n", del, index, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del, d.marks.history, del);

printBorder();

printf("\n--> Выход из редактирования\n");

break;

}

default: { }

}

COORD positionCur = { 50,3 }; //позиция x и y

c = getch();

if (c != 13) {

position = GetCurentSelector(c, position, 2, 11);

if ((position[1] == 2) && (!flag)) {

tempPosition = position[0];

position[0] = 11;

flag = 1;

}

else { if (flag) { position[0] = tempPosition; flag = 0; } }

}

else {

switch (position[0]) {

case 1:

case 2: {

positionCur.X = 7; //x

SetConsoleCursorPosition(hConsole, positionCur);

printf(" ");

SetConsoleCursorPosition(hConsole, positionCur);

scanf("%d", &d.id);

break;

}

case 3: {

positionCur.X = 16; //x

SetConsoleCursorPosition(hConsole, positionCur);

printf(" ");

SetConsoleCursorPosition(hConsole, positionCur);

scanf("%s", &d.surname);

break;

}

case 4: {

positionCur.X = 51; //x

SetConsoleCursorPosition(hConsole, positionCur);

printf(" ");

SetConsoleCursorPosition(hConsole, positionCur);

scanf("%s", &d.name);

break;

}

case 5: {

positionCur.X = 84; //x

SetConsoleCursorPosition(hConsole, positionCur);

printf(" ");

SetConsoleCursorPosition(hConsole, positionCur);

scanf("%d", &d.birth\_date);

break;

}

case 6: {

positionCur.X = 101; //x

SetConsoleCursorPosition(hConsole, positionCur);

printf(" ");

SetConsoleCursorPosition(hConsole, positionCur);

scanf("%d", &d.stud\_date);

break;

}

case 7: {

positionCur.X = 121; //x

SetConsoleCursorPosition(hConsole, positionCur);

printf(" ");

SetConsoleCursorPosition(hConsole, positionCur);

scanf("%d", &d.marks.fiz);

break;

}

case 8: {

positionCur.X = 132; //x

SetConsoleCursorPosition(hConsole, positionCur);

printf(" ");

SetConsoleCursorPosition(hConsole, positionCur);

scanf("%d", &d.marks.math);

break;

}

case 9: {

positionCur.X = 147; //x

SetConsoleCursorPosition(hConsole, positionCur);

printf(" ");

SetConsoleCursorPosition(hConsole, positionCur);

scanf("%d", &d.marks.it);

break;

}

case 10: {

positionCur.X = 168; //x

SetConsoleCursorPosition(hConsole, positionCur);

printf(" ");

SetConsoleCursorPosition(hConsole, positionCur);

scanf("%d", &d.marks.history);

break;

}

case 11: {

return d;

break;

}

}

}

}

}

else {

printf("\n");

printTable();

printf("| | "); scanf("%d", &d.id); clear();

printTable(); printf("| | %-6d | ", d.id);

scanf("%s", &d.surname); clear(); char del = '|'; int l = 1;

printTable(); printf("%c %-3d %c %-6d %c %-32s %c", del, l, del, d.id, del, d.surname, del);

scanf("%s", &d.name);

clear(); printTable();

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c", del, l, del, d.id, del, d.surname, del, d.name, del);

scanf("%d", &d.birth\_date);

clear(); printTable();

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c", del, l, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del);

scanf("%d", &d.stud\_date);

clear(); printTable();

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c", del, l, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del);

scanf("%d", &d.marks.fiz);

clear(); printTable();

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c %-8d %c", del, l, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del);

scanf("%d", &d.marks.math);

clear(); printTable();

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c %-8d %c %-12d %c", del, l, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del);

scanf("%d", &d.marks.it);

clear(); printTable();

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c %-8d %c %-12d %c %-18d %c", del, l, del, d.id, del, d.surname, del, d.name, del, d.birth\_date, del, d.stud\_date, del, d.marks.fiz, del, d.marks.math, del, d.marks.it, del);

scanf("%d", &d.marks.history);

}

return d;

}

int printBorder() {

for (int i = 0; i < 180; i++)

printf("-");

}

int printTable()

{

char\* TableColsName[] = { " № "," id "," Фамилия "," Имя "," Год рождения "," Год поступления "," Физика "," Математика "," Программирование "," История " };

printBorder(); printf("\n"); char del = '|';

printf("%c %s %c %s %-13c %-8s %12c %18s %13c %s %c %s %c %s %c %s %c %s %c %s %c\n", del, TableColsName[0], del, TableColsName[1], del, TableColsName[2], del, TableColsName[3], del, TableColsName[4], del, TableColsName[5], del, TableColsName[6], del, TableColsName[7], del, TableColsName[8], del, TableColsName[9], del);

printBorder(); printf("\n");

}

void printINFO(STudent\* d, int index)

{

if (index == 1 || index == 0) printTable(); //заголовок таблицы

char del = '|';

printf("%c %-3d %c %-6d %c %-32s %c %-30s %c %-14d %c %-17d %c %-8d %c %-12d %c %-18d %c %-9d %c", del, index, del, d->id, del, d->surname, del, d->name, del, d->birth\_date, del, d->stud\_date, del, d->marks.fiz, del, d->marks.math, del, d->marks.it, del, d->marks.history, del);

printf("\n");

printBorder();

printf("\n");

}

int correctInfo(Person\* st)

{

char\*\* Menu1[] = { "Найти по номеру","Выйти в главное меню", "Редактировать","Сохранить" }; //меню если элемент для редактирования активен

char\*\* Menu2[] = { "Найти по номеру","Выйти в главное меню" }; //меню если элемента нет

int position[] = { 1,1 }; //массив позиции для меню

STudent d, old; d.id = NULL; d.birth\_date = NULL; Person\* temp = NULL;

while (1)

{

int select; int Posid = 1; clear();

if ((d.id != NULL) || d.birth\_date != NULL) //если запись с которо работаем - есть

{

select = PrintMenuWithTable(Menu1, position, 4, 1, printINFO, &d, 1); // вызов меню с чермя параметрами

}

else select = PrintMenu(Menu2, position, 2, 1); // если записи нет - меню с двумя параметрами

switch (select)

{

case 3:

{

d = GetInfoFromCeyboard(d); // получение данных с клавиатуры

break;

}

case 4:

{

clear();

printf("\n--------------------------------------------------------------------------------- Новая запись ------------------------------------------------------------------------------------- \n");

printINFO(&d, 1); // вывод информации

printf("\n-------------------------------------------------------------------------------- Старая запись ------------------------------------------------------------------------------------- \n");

printINFO(&old, 1); // вывод информации

printf("\n");

if (getResponse()) {

temp->info = d;

puts("Данные сохранены . Нажмите любую кнопку");

getch();

}

}

break;

case 1: {

int SerchID; printf("\nВведите идентификационный номер -->"); scanf("%d", &SerchID);

int flag = 0; if (!st) break;

temp = getLeaf(st, SerchID);

if (temp) {

d = temp->info;

old = temp->info;

}

if (!temp) {

printf("Не найден студент с таким идентификатором .... Нажмите любую кнопку"); getch();

d.id = NULL; d.birth\_date = NULL;

}

break; }

case 2: return 0;

}

}

}

Person\* getLeaf(Person\* root, int index)

{

if (root == NULL)

return NULL;

else if (root->info.id == index)

return root;

else if (root->info.id < index)

return getLeaf(root->right, index);

else

return getLeaf(root->left, index);

}

**4. Результаты тестирования и отладки программы**

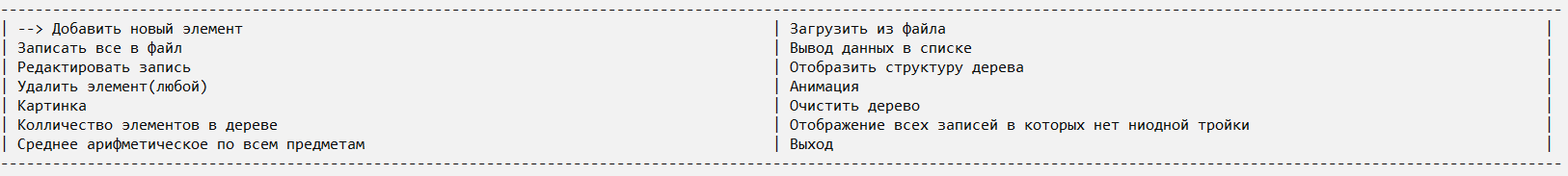


Рисунок 10 – Меню выбора действий



Рисунок 11 – Меню добавления записи

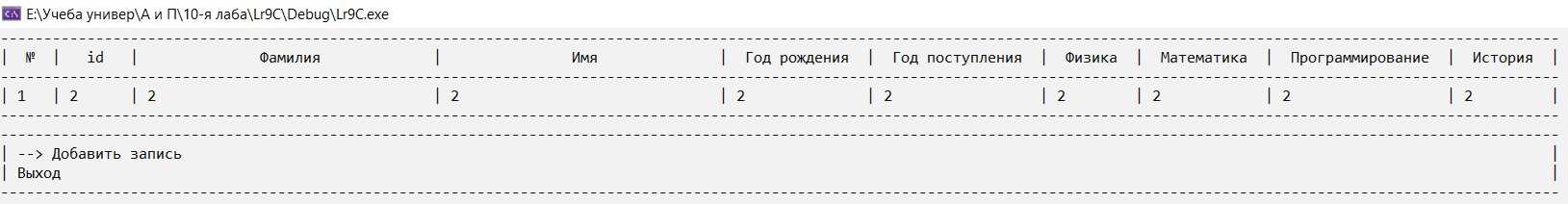


Рисунок 12 – Форма после добавления записи через нее

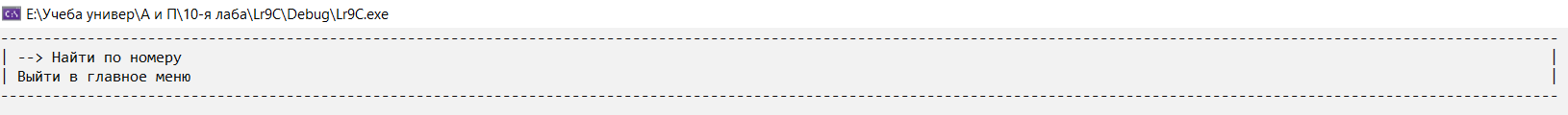


Рисунок 13 – Меню поиска и редактирования записей

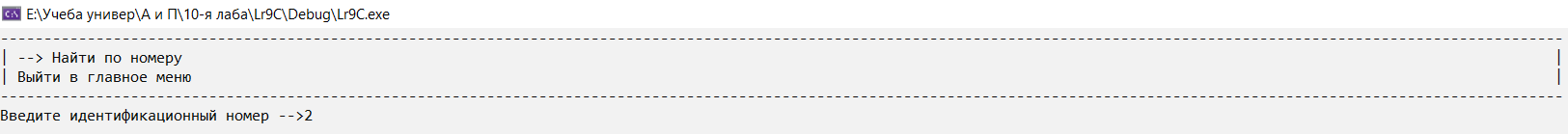


Рисунок 14 – Реализация поиска по номеру

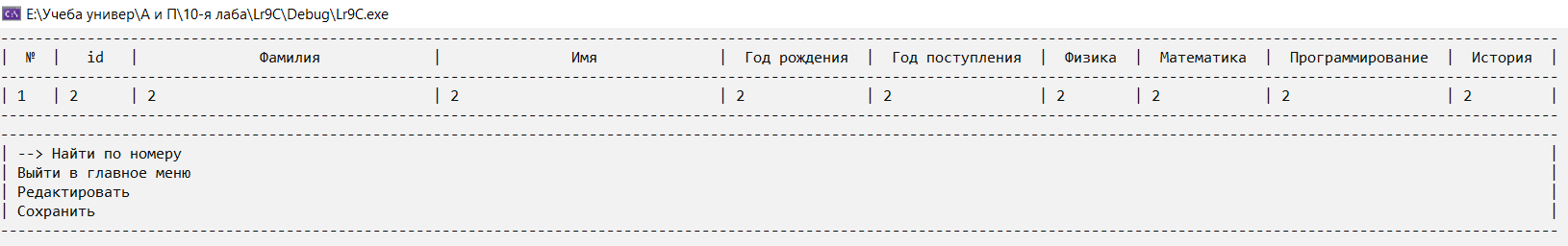


Рисунок 15 – Меню после после того как запись с таким номером найдена

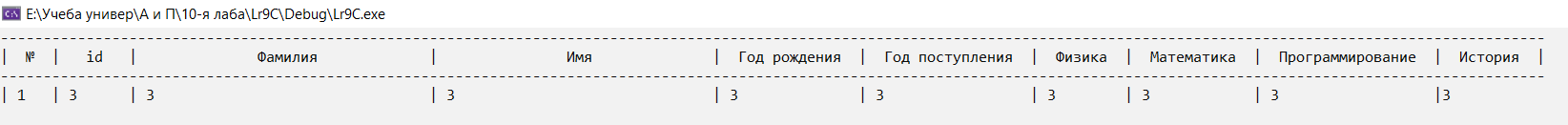


Рисунок 16 – Редактирование записи

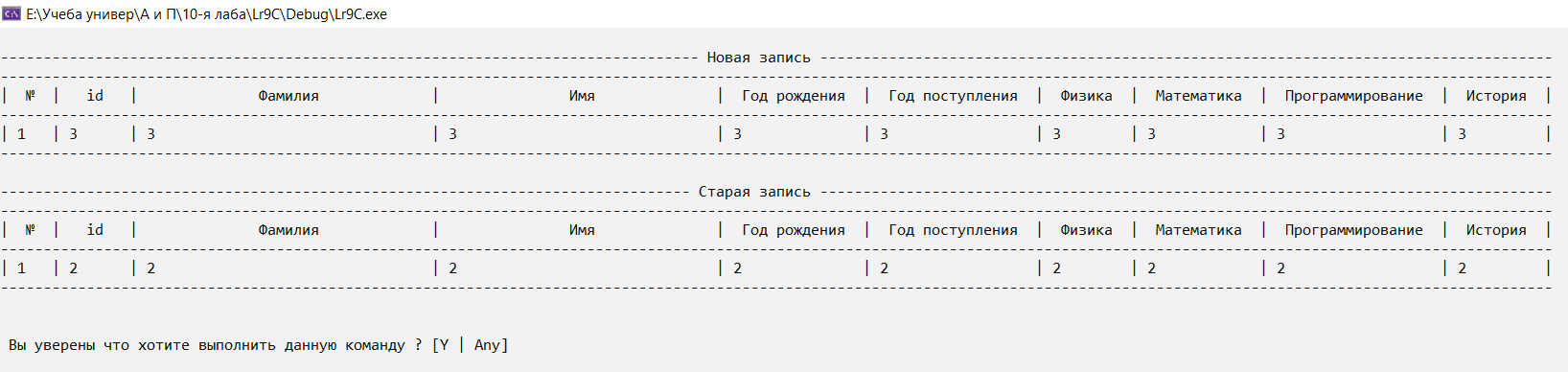


Рисунок 17 – Сохранение записи.

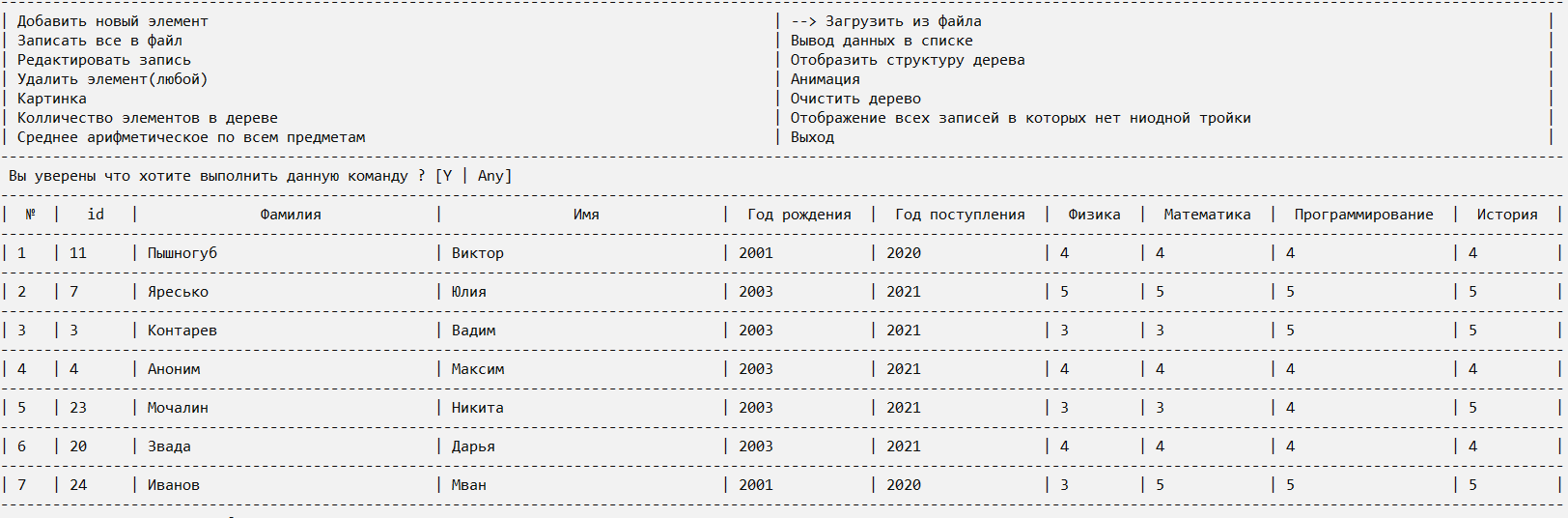


Рисунок 18 – Загрузка данных из файла

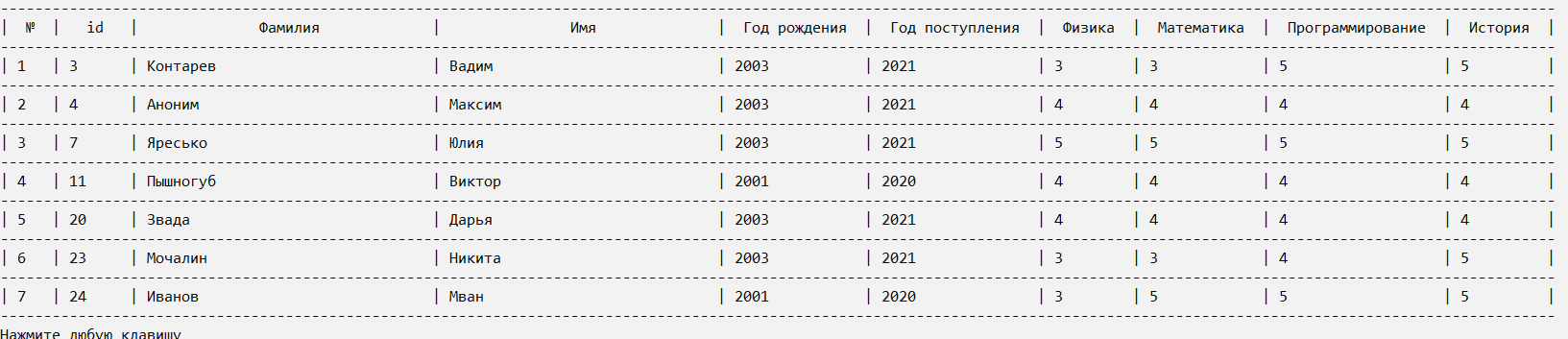


Рисунок 19 – Вывод всех элементов в дереве на экран

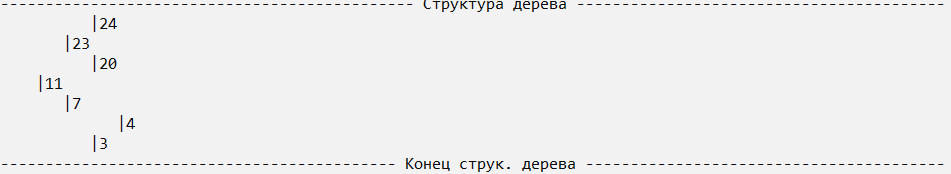


Рисунок 20 – Структура дерева в памяти

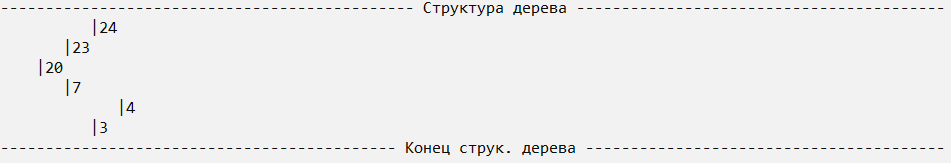
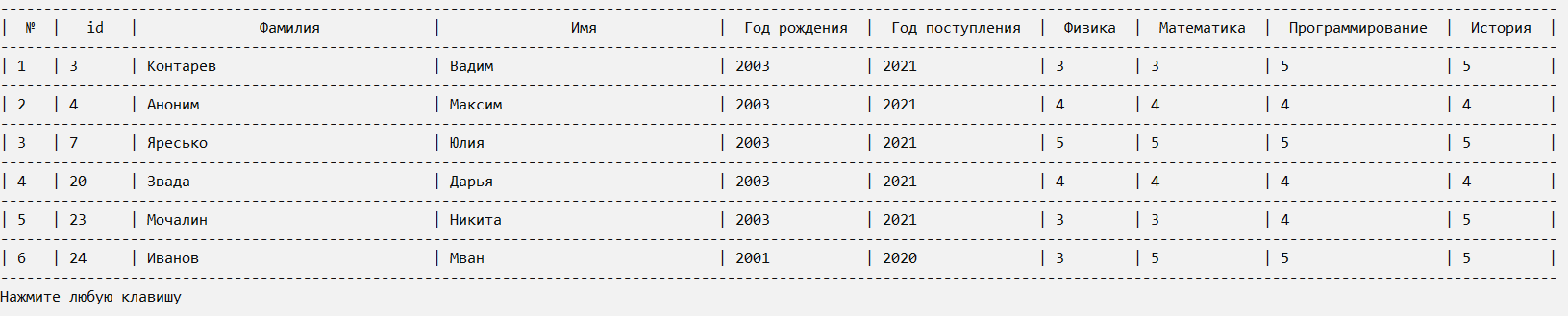
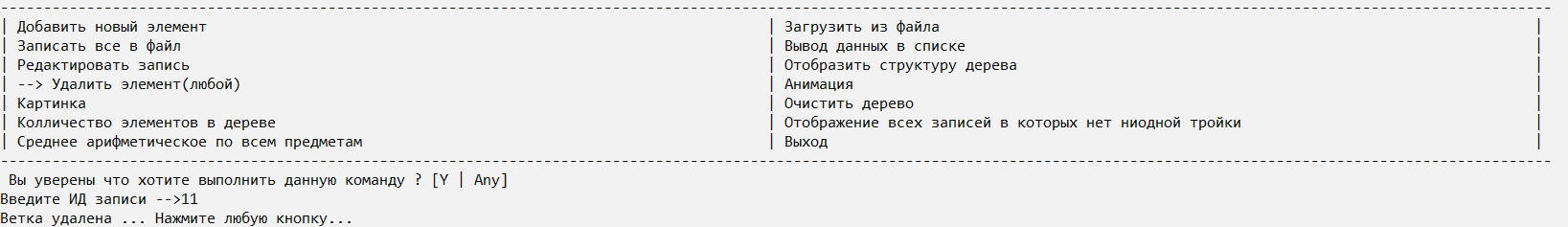
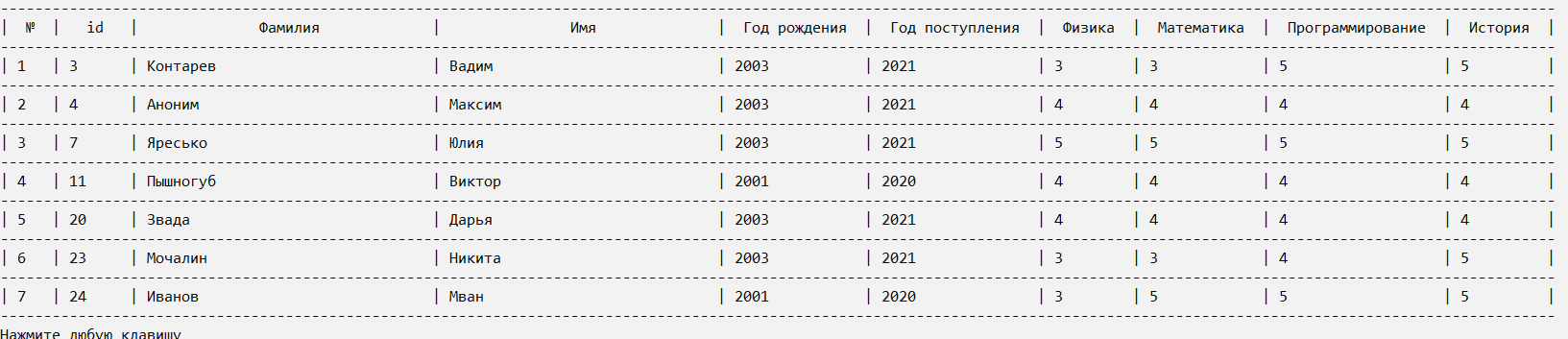
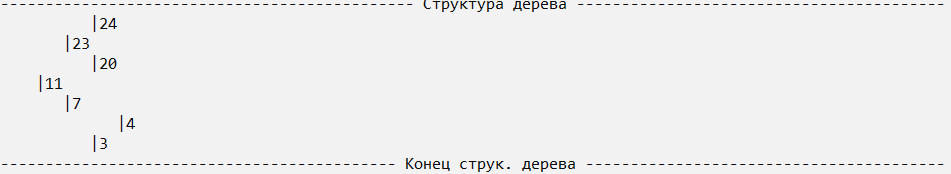


Рисунок 21 – Удаление записи из дерева по ид.

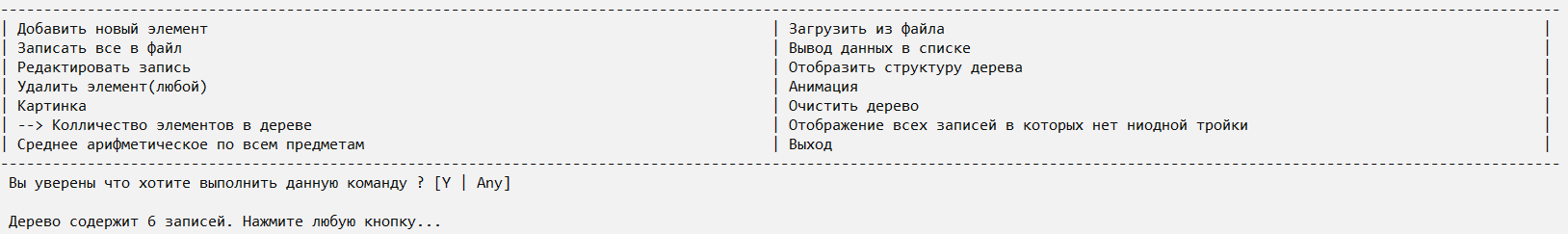


Рисунок 22 – Количество элементов в дереве.

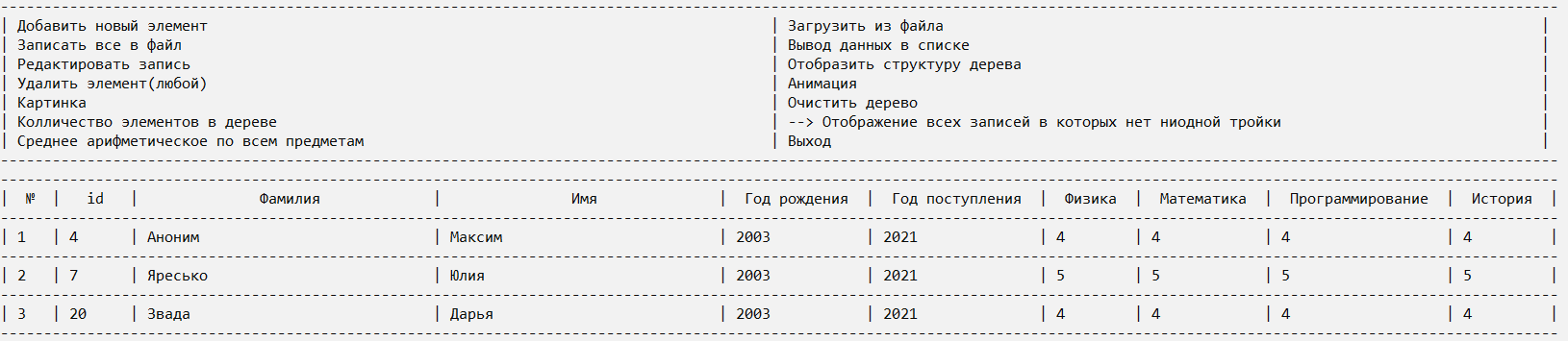


Рисунок 23 – Все студенты у которых нет троек.

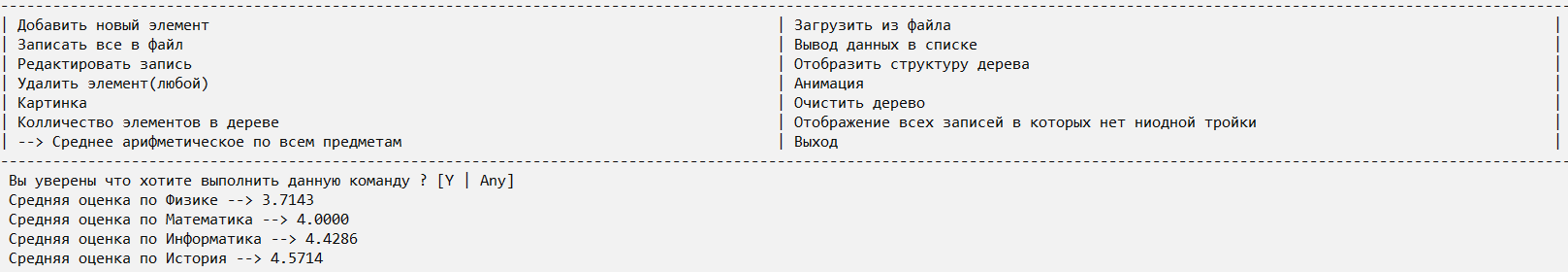


Рисунок 24 – Среднее арифметическое по всем предметам

**Вывод**

В ходе выполнения данной лабораторной работы были изучены основные операции с модулями в языках C/С++, использование стражей включения и передача каких либо параметров между модулями.