**ЛАБОРАТОРНАЯ РАБОТА №4**

**«ПРОГРАММИРОВАНИЕ НЕЛИНЕЙНЫХ**

**СТРУКТУР ДАННЫХ НА ЯЗЫКЕ С/С++»**

**Цель работы**: Изучение нелинейных структур данных и приобретение навыков разработки и отладки программ, использующих древовидные структуры данных. Исследование особенностей работы с поисковыми бинарными деревьями на языке 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.

**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**

#pragma warning(disable : 4996) ;

#pragma warning(disable : 6031) ;

#include <stdio.h>

#include <string.h>

#include <conio.h>

#include <windows.h>

#include <stdlib.h>

#include "C:\Users\Виктор\source\repos\VicMenuDLL\VicMenuDLL\VicMenuDLL.cpp"; //файл библиотеки меню

#define clear() system("cls")

//добавить новый так, чтобы сохранилась упорядоченность

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;

typedef struct Persons {

STudent info;

struct Persons\* right;

struct Persons\* left;

} Person;

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;

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

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

void addToTree(Person\*\* head, const STudent\* info);//функция добавления в начало списка

int аddNewElement(Person\*\* st);// Функция добавления элемента в конец

STudent GetInfoFromCeyboard(STudent d); //форма считывания с клавиатуры данных

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

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

int create\_file(FILE\* f, Person\* St); //Функция записи спсика в файл

Person\* loadFromFile\_new(FILE\* f); //Функция загрузки данных из файла

void printToFile(FILE\* f, Person\* St);//Функция добавления элемента в файл

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

int correctInfo(Person\* f); //Редактирование записи по ид

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

Person\* deleteThree(Person\* st);//удалить дерево

Person\* searchElementById(Person\*, int); //Функция поиска элемента по id , возвращает ссылку (положение)

Person\* DeleteNode(Person\* root, int id);//Функция удаления записи из дерева. Удалить можно любую запись

int getLeafCount(Person\* root, int count);//Подсчет листьев в дереве

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

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

void printTable(); //Напечатать заголовок таблицы

void printBorder(); //Вертекальный разделитель таблицы

int PrintTreeData(Person\* st); //вывод дерева на экран

void printINFO(STudent d, int index);//Вывод одной записи на экран

int PrintTreeDataNonThree(Person\* root);//вывод студентов без 3)

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

void MenuSelect(int selector, FILE\* f); //Основная функция, которая отвечает за запуск функций в зависимости от полученного значения селектора

int getResponse();//Получить подтверждение от пользователя

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

Person\* st = NULL;

Person\* root = NULL;

int position[] = { 1,1 };

int main(void) {

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;

}

}

void printBorder() {

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

printf("-");

}

void 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 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\* searchElementById(Person\* head, int id)

{

Person\* elem = NULL;

if (head)

{

for (Person\* tmp = head; tmp; tmp = tmp->right)

if (tmp->info.id == id)

{

elem = tmp;

break;

}

}

return elem;

}

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;

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 = 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\* 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);

}

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;

}

}

}

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)

{

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;

}

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;

} **4. Результаты тестирования и отладки программы**

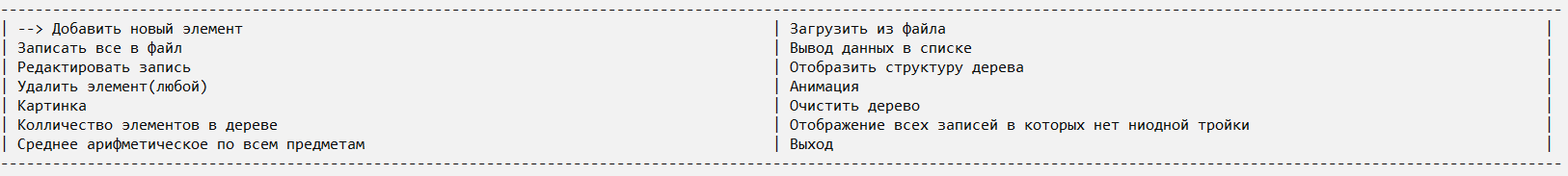


Рисунок 10 – Меню выбора действий



Рисунок 11 – Меню добавления записи

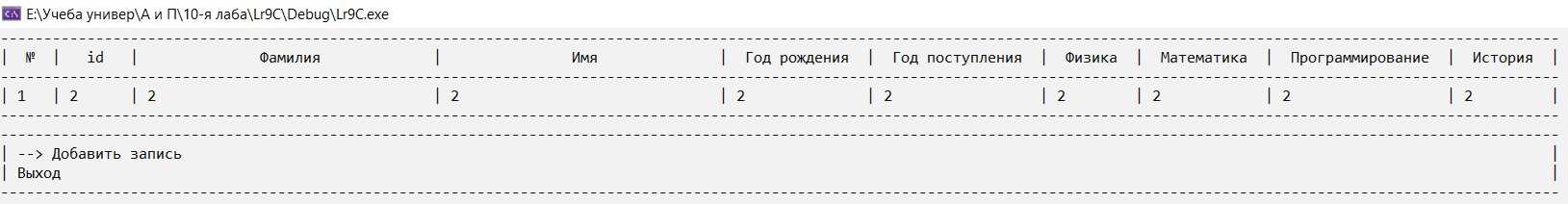


Рисунок 12 – Форма после добавления записи через нее

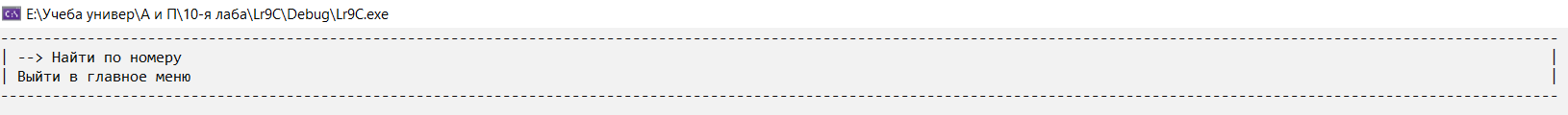


Рисунок 13 – Меню поиска и редактирования записей

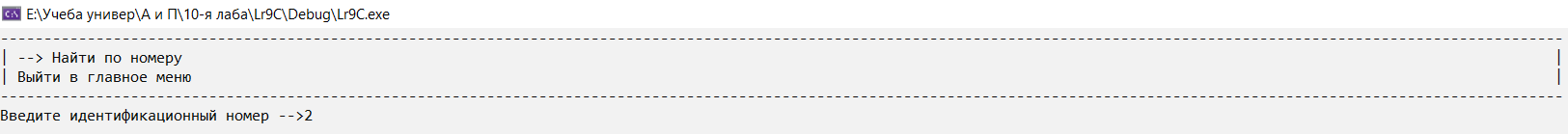


Рисунок 14 – Реализация поиска по номеру

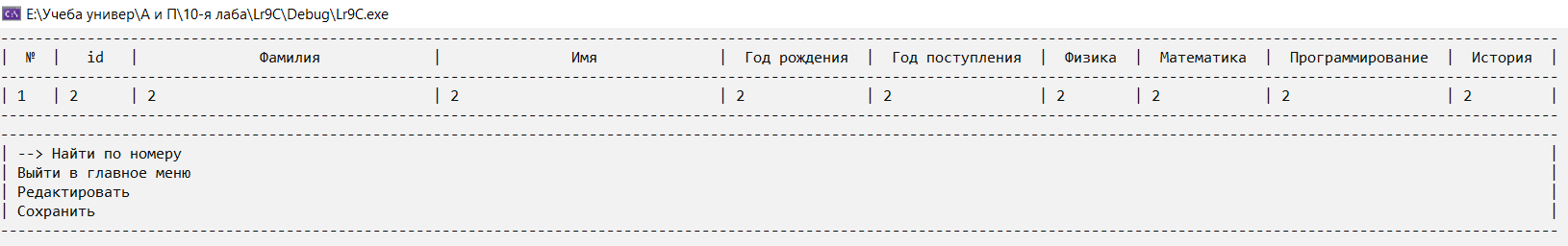


Рисунок 15 – Меню после после того как запись с таким номером найдена

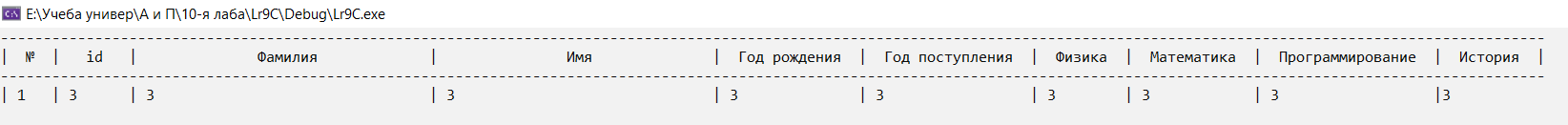


Рисунок 16 – Редактирование записи

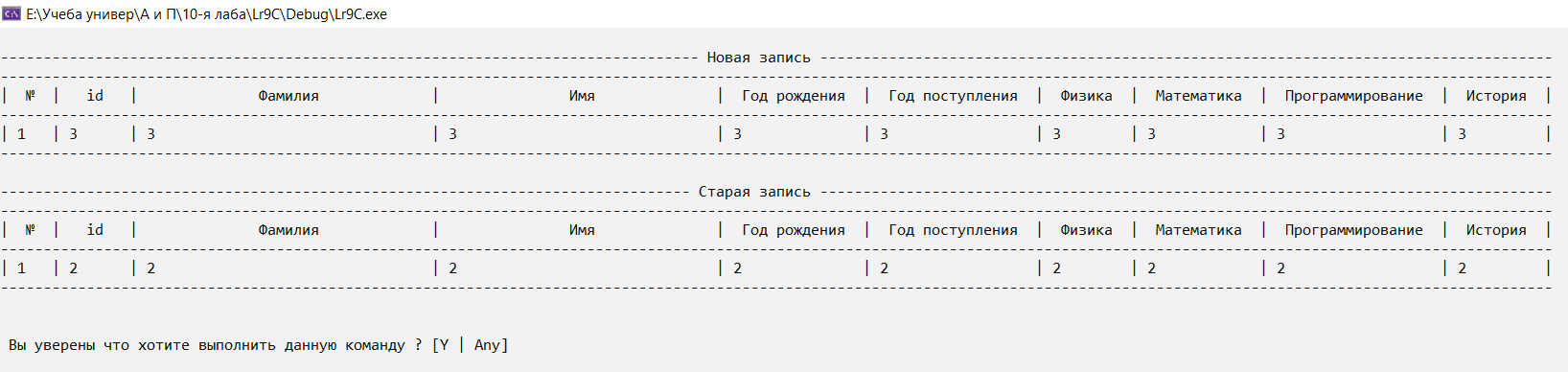


Рисунок 17 – Сохранение записи.

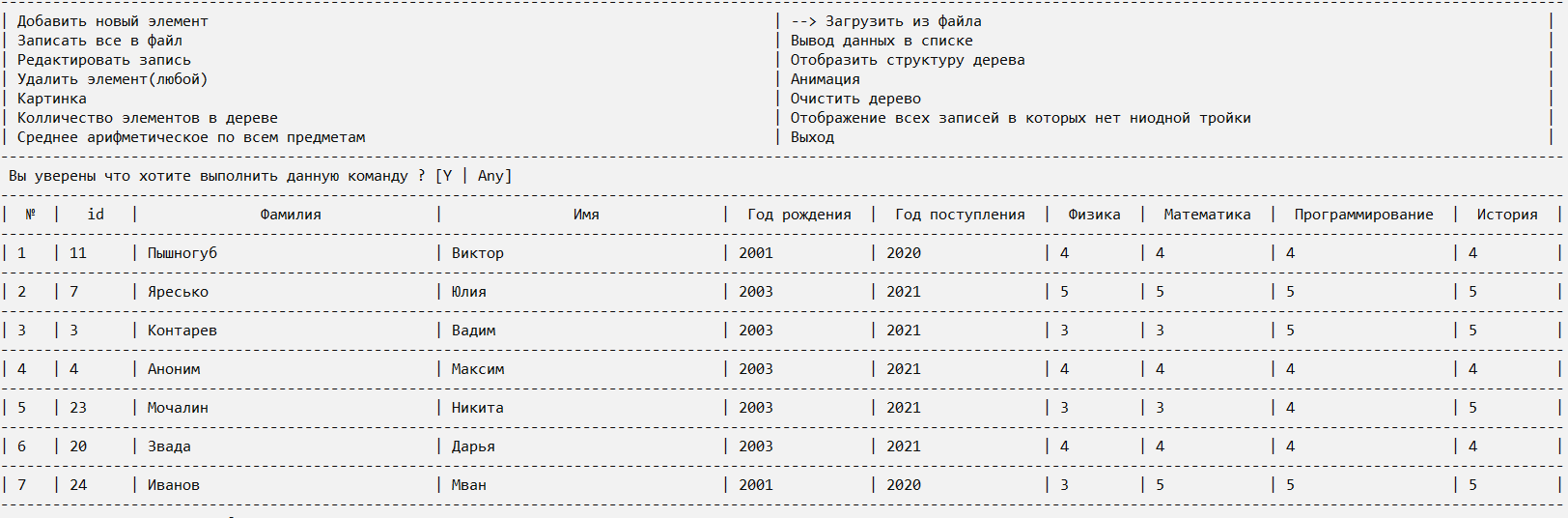


Рисунок 18 – Загрузка данных из файла

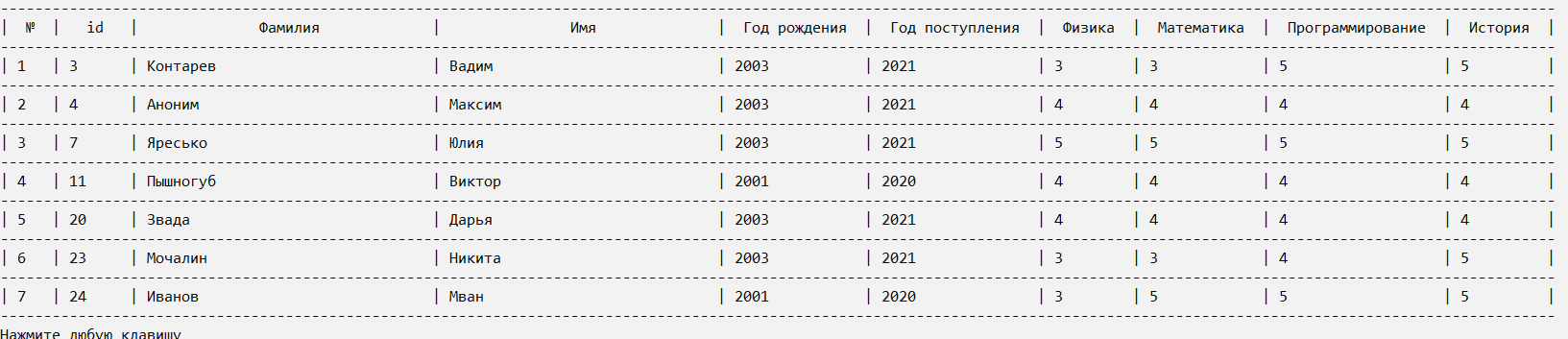


Рисунок 19 – Вывод всех элементов в дереве на экран

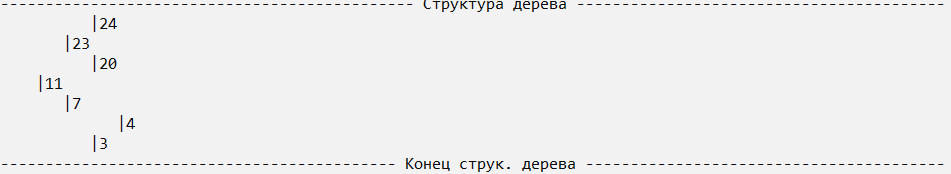


Рисунок 20 – Структура дерева в памяти

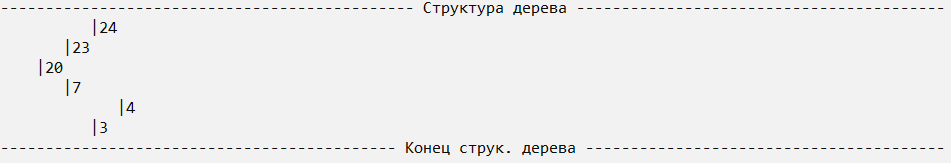
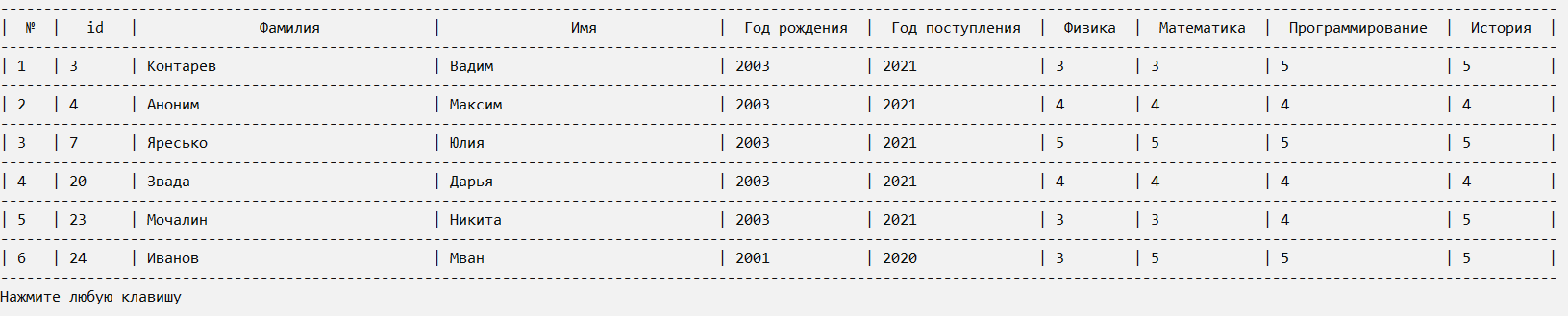
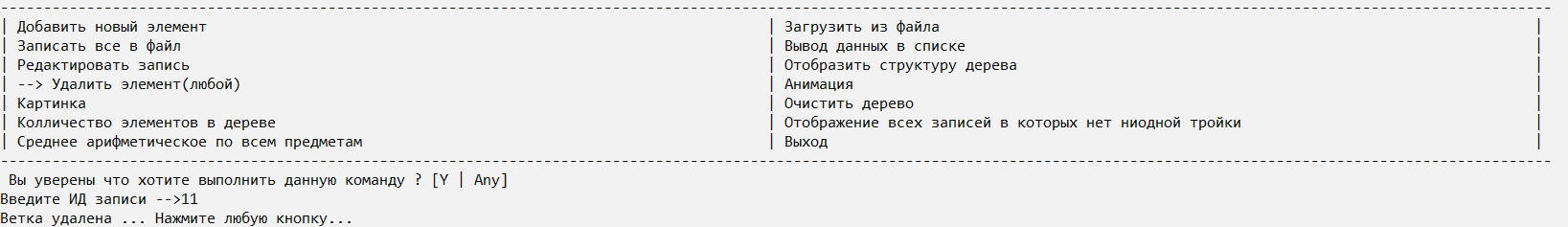
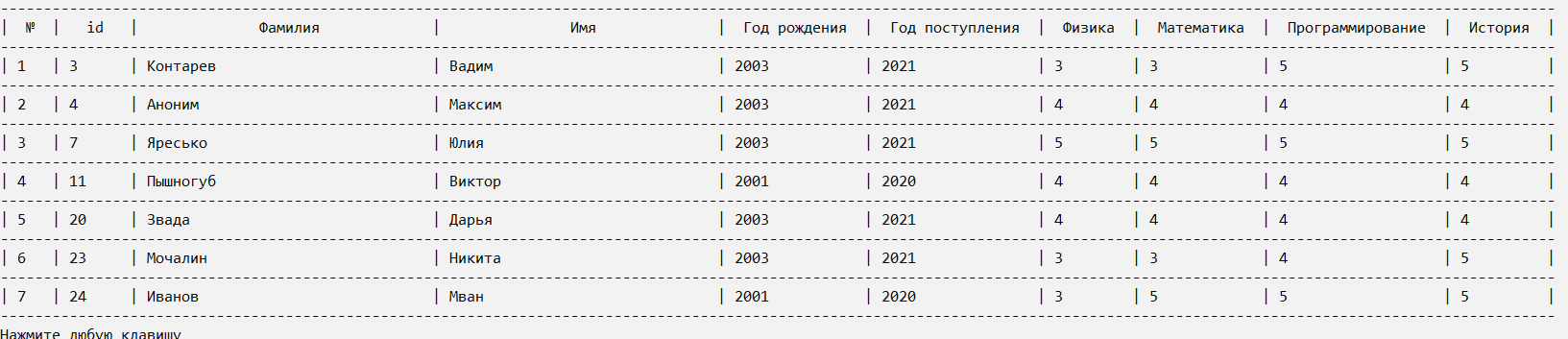
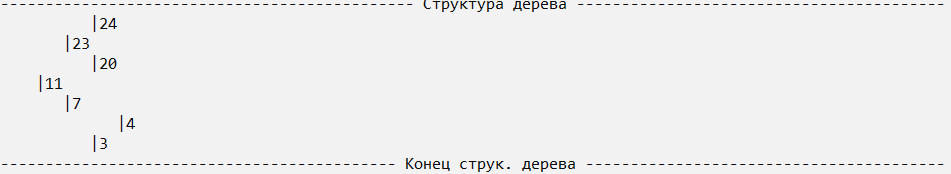


Рисунок 21 – Удаление записи из дерева по ид.

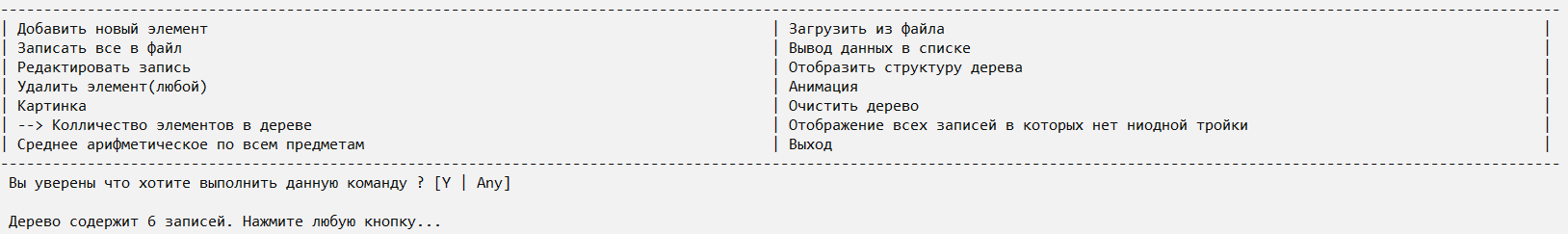


Рисунок 22 – Количество элементов в дереве.

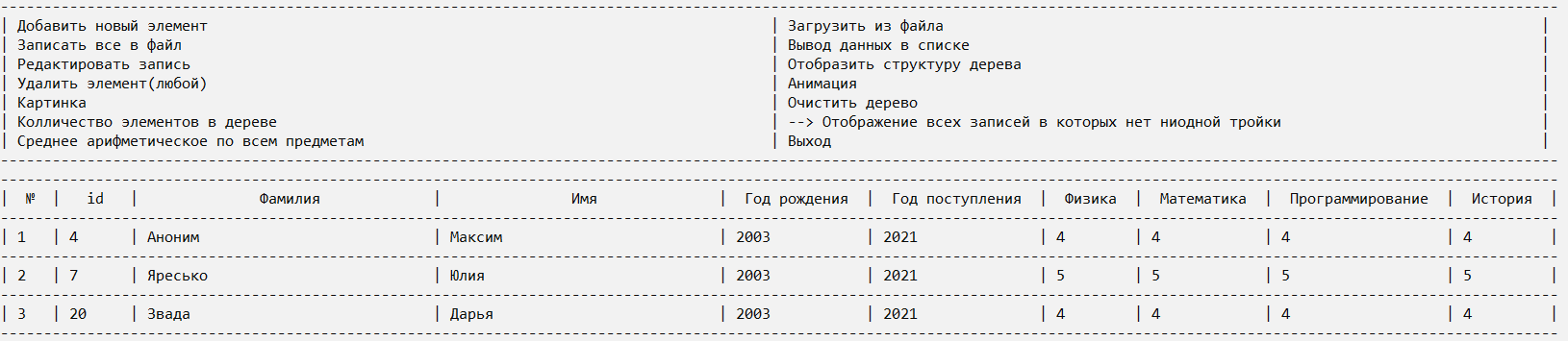


Рисунок 23 – Все студенты у которых нет троек.

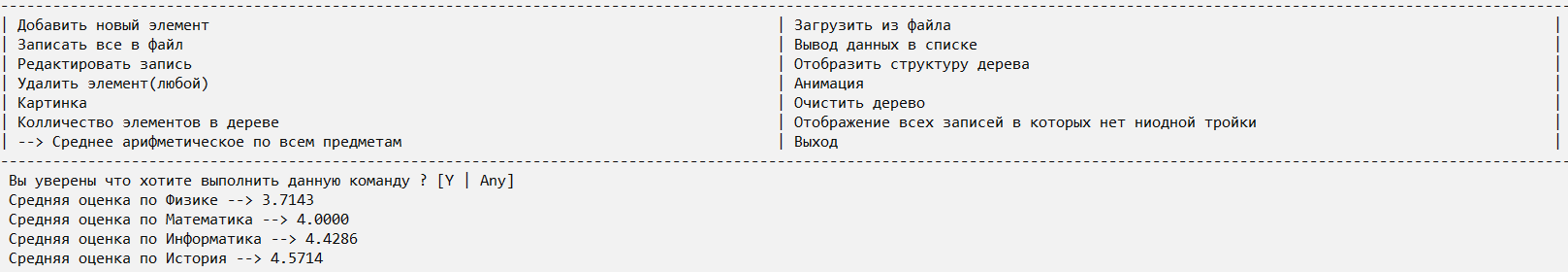


Рисунок 24 – Среднее арифметическое по всем предметам

**Вывод**

В ходе выполнения данной лабораторной работы были изучены основные операции над бинарными деревьями, принцип работы с деревьями и указателями на элемент, обработки бинарной информации полученной из файла в языке С, с учетом связи «Родитель 🡪 потомок». Получены практические навыки работы с файловыми указателями, бинарными деревьями, реализации алгоритмов обработки данных на прямую в узлах дерева на языке С. Реализован принцип минималистического графического меню для работы в программе, написана своя библиотека.