МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВОЙ ПРОЕКТ

ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

доц., к.т.н А.В. Туманова должность, уч. степень звание подпись, дата инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

**РАЗРАБОТКА ПРОГРАММЫ «ИНФОРМАЦИЯ О СОТРУДНИКАХ»**

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. Z7431 30.12.2018 М.Д.Семочкин

подпись, дата инициалы, фамилия

Санкт-Петербург

2018

**1. Постановка задачи**

Задачей курсового проекта является разработка программы для хранения информации о сотрудниках, которая позволяет вводить информацию, хранить ее в файле, осуществлять поиск, модификацию, сортировку и удаление данных.

В соответствии с вариантом 16:

Данные о человеке хранятся в структуре с именем ZNAK, содержащей следующие поля:

•  фамилия, имя;

•  знак зодиака;

•  дата рождения (массив из трех чисел).

Задание на поиск: найти информацию о людях, родившихся в месяц, значение которого введено с клавиатуры.

**2. Описание структур данных**

1) Данные о человеке хранятся в структуре с именем ZNAK

**struct** ZNAK { *// структура ZNAK*

**char** name[NAME\_LENGTH]; *// имя*

**char** zodiaс[ZODIAC\_LENGTH]; *// знак зодиака*

**int** birthDate[3]; *// дата рождения*

};

Имена сотрудников в программе являются уникальными. По ним производится поиск человека для редактирования или удаления записи.

2) Размеры массивов для записи имени и знака зодиака хранятся в константах:

**const** **int** NAME\_LENGTH = 512;

**const** **int** ZODIAC\_LENGTH = 18;

В случае ввода строки длиннее максимальной допустимой длины строка обрезается при записи в структуру.

Минимальное допусимое значение года рождения также хранится в константе:

**const** **int** MIN\_ALLOWED\_BIRTH\_YEAR = 1900;

3) База данных хранится на диске в виде текстового файла.

Формат записи:

|  |
| --- |
| Антон Сергеевич Дубов 6/5/1938 Весы  Егор Сергеевич Иванченко 1/2/1980 Близнецы |

4) Для работы с БД в процессе выполнения программы используется однонаправленный связный список:

**struct** List { *// связный список для хранения записей*

ZNAK data;

List\* next;

};

**3. Описание программы и созданных функций**

1. **void** printRecord(ZNAK record) *// Вывод записи*
2. **void** ShowList(List \*begin) *// Вывод всех элементов списка на экран*
3. **void** AddElem(List \*\*begin, List \*\* cur, ZNAK elem) *// Добавление элемента в конец списка*
4. **void** AddFirstElem(List \*\*begin, ZNAK elem) *// Добавление элемента в начало списка*
5. List \* FindElem(List \*begin, ZNAK elem) *// поиск элемента в списке по имени*
6. **void** FindElem(List \*begin, **int** month) *// поиск элемента в списке по месяцу рождения и его вывод*
7. **void** DelElem(List \*\*begin, List\* ptr, List \*\* cur) *// Удаление элемента из списка*
8. **void** Free(List \*\*begin) *// Очистка памяти*
9. **void** calculateZodiac(ZNAK & record) *// Определение знака зодиака по дате*
10. **bool** leapYear(**int** year) *// Проверка является ли год високосным*
11. **void** inputData(List \*begin, ZNAK & record) *// Ввод данных*
12. **void** inputName(ZNAK & record) *// Ввод имени для поиска записи*
13. **void** saveToFile(List \*begin) *// Запись в файл*
14. **void** readLineFromFile(string& inputLine, List \*begin, ZNAK & record) *// Парсинг строки из файла*
15. **void** readFromFile(List \*\*head, List \*begin, List \*\*cur, ZNAK & record) *// Чтение из файла*

**4. Описание пользовательского интерфейса**

Программа реализована на языке С++ в виде консольного приложения.

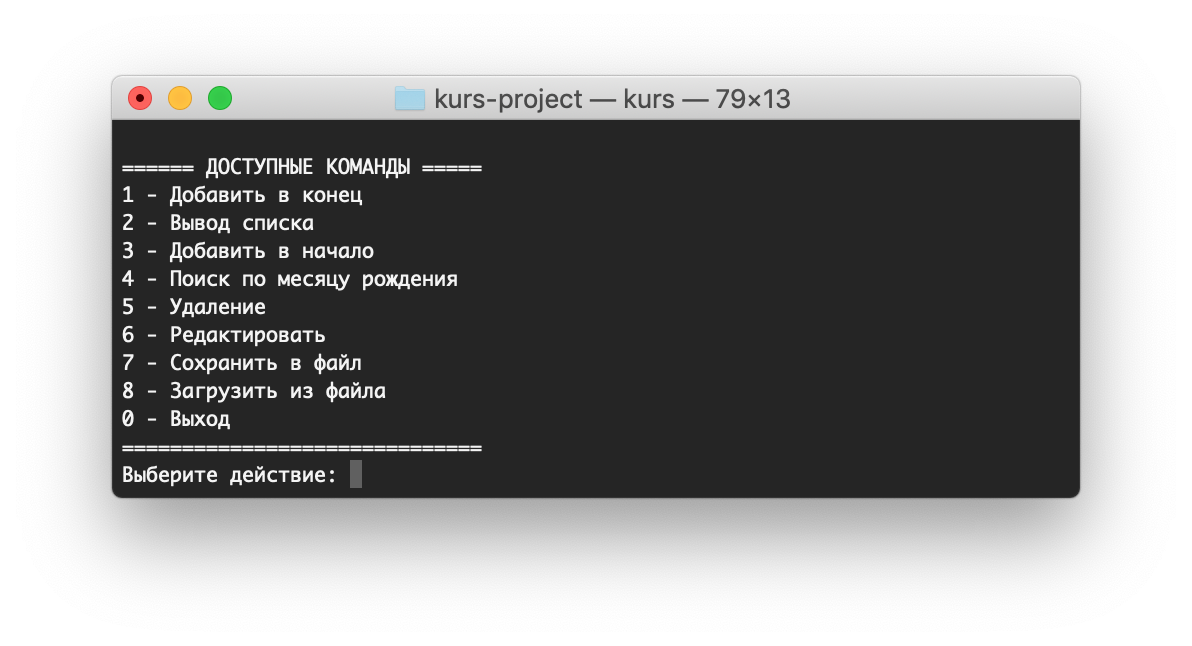
В главной функции main() реализовано меню пользователя, в котором каждому действию соответствует определенная цифра. Введя цифру 0, пользователь может выйти из программы.

Реализованы следующие функции для работы с данными:

* добавление записи в конец списка
* вывод списка на экран
* добавление записи в начало списка
* поиск по месяцу рождения
* удаление записи
* редактирование записи
* сохранение в файл

Поиск записи для редактирования и удаления производится по имени человека.

Имя является уникальным идентификатором записи.



4.1 Добавление записи в конец списка

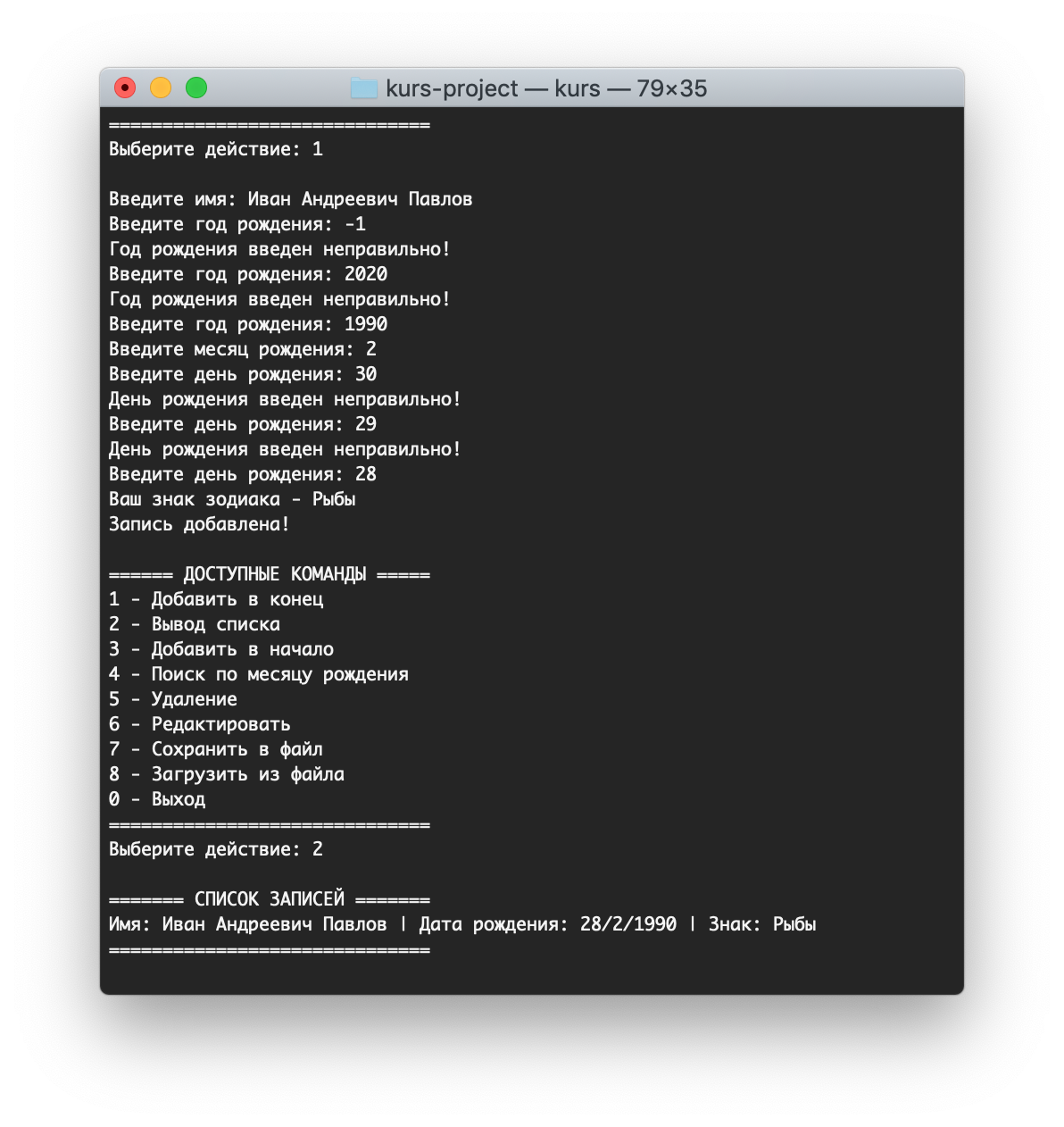
Функция выполняет добавление записи в список. При вводе данных выполняется проверка формата введенных данных:

- дата рождения:

* год должен быть больше значения константы MIN\_ALLOWED\_BIRTH\_YEAR
* год должен быть меньше или равен текущему году
* месяц должен быть числом от 1 до 12
* в случае если введен текущий год, номер введенного месяца не может быть больше номера текущего месяца
* день должен быть числом от 1 до максимально возможного номера дня во введенном месяце (учитываются високосные года)

Если введены некорректные данные, пользователю предлагается ввести из повторно. Также выполняется проверка на уникальность имени.

Если пользователь ввел не уникальное имя, будет предложено ввести имя повторно.

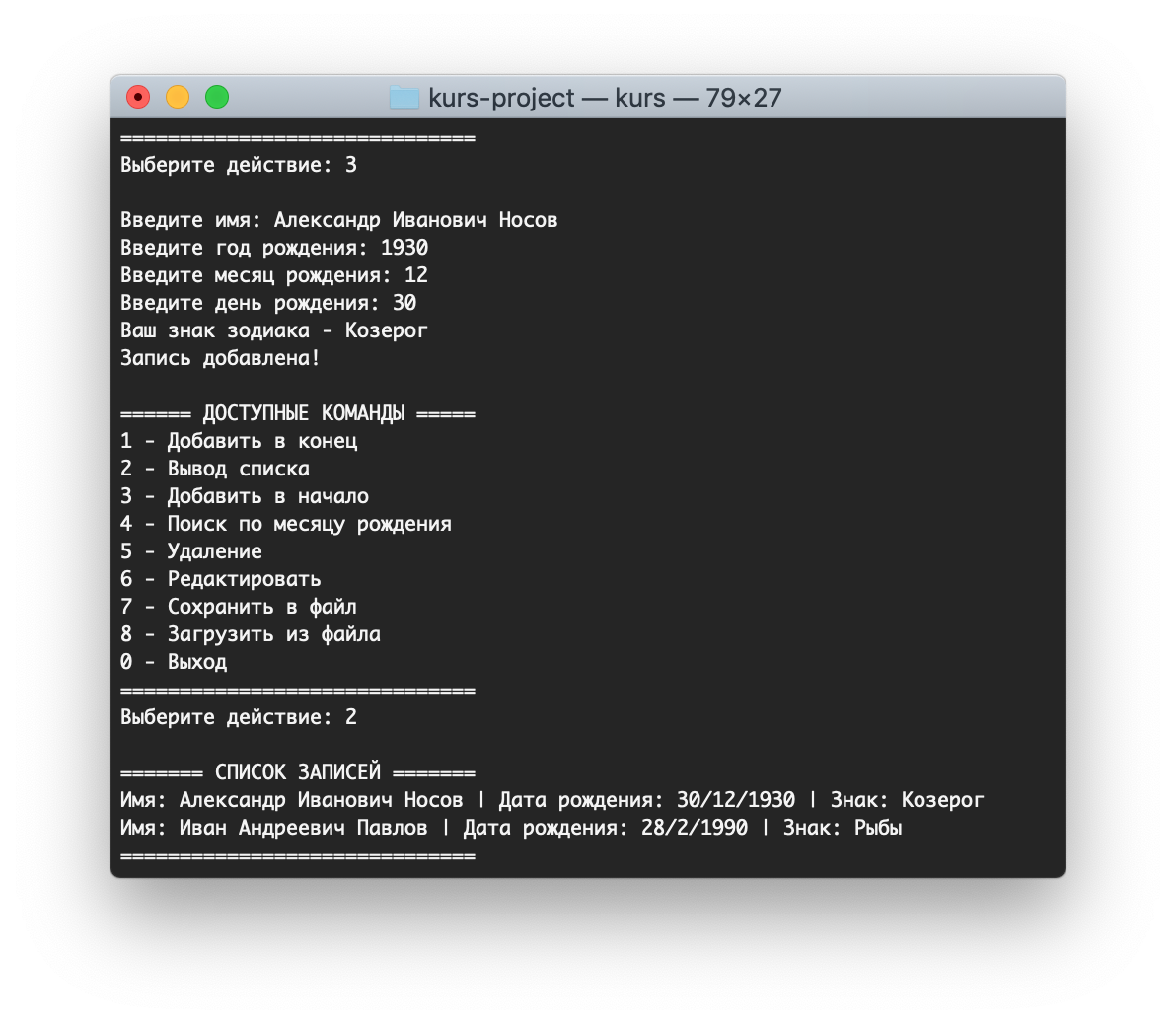


4.2 Вывод списка

Функция выводит все записи на экран. Результат работы можно увидеть на скриншоте с демонстрацией работы функции «Добавить в конец»

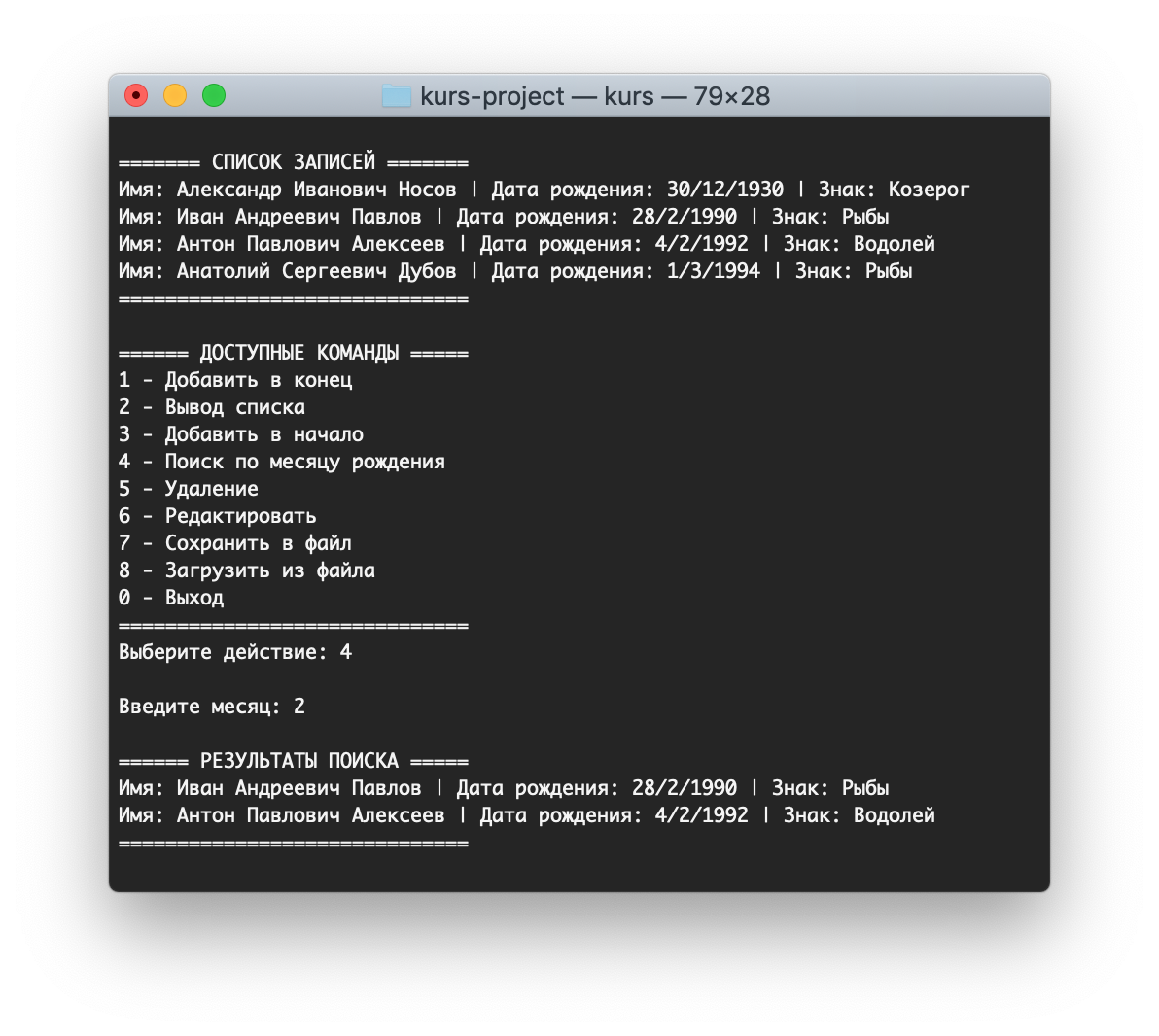
4.3 Добавление записи в начало списка

Функция работает аналогично функции «Добавить в конец», но добавляет запись в конец списка:



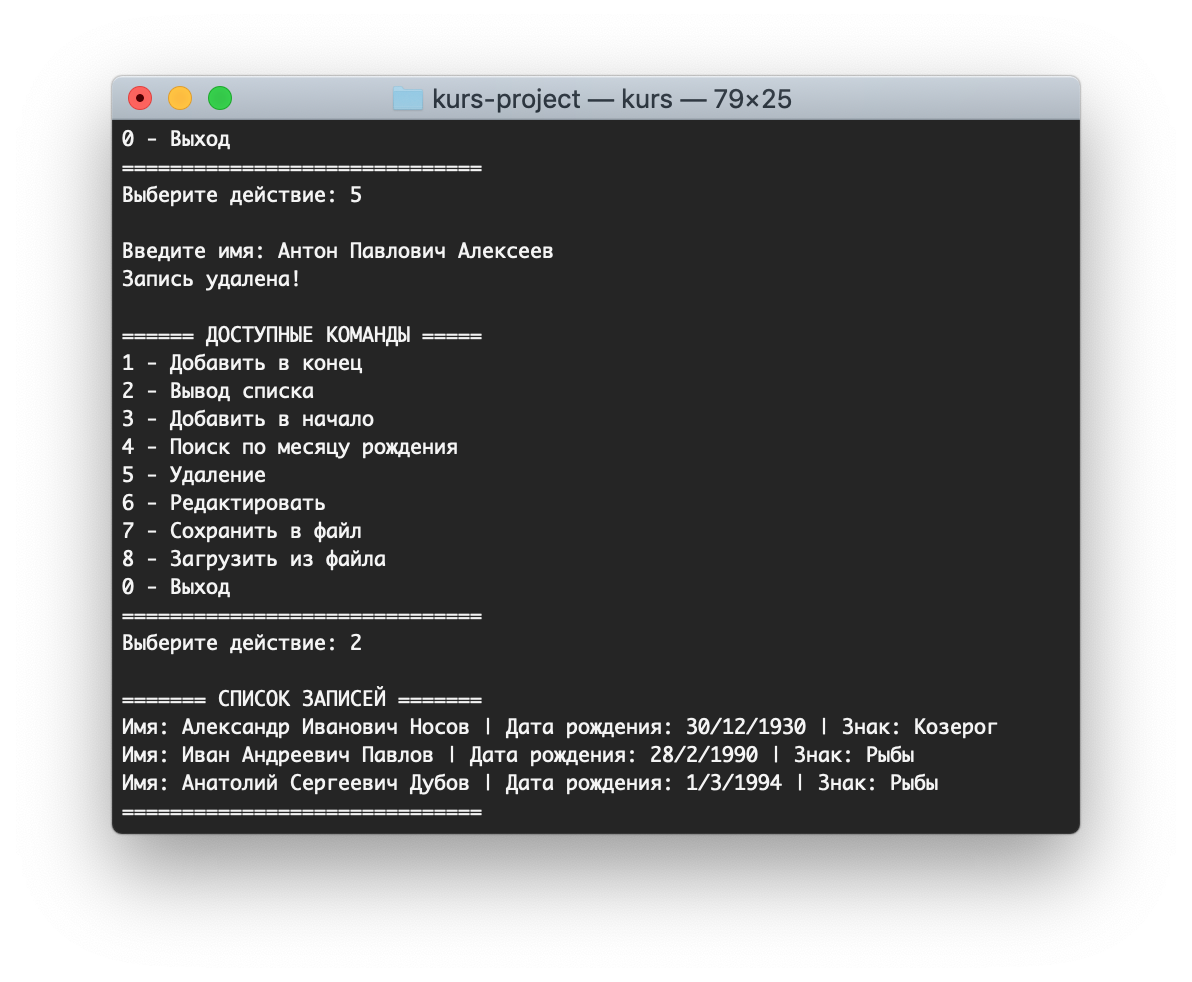
4.4 Поиск по месяцу рождения:

Функция выводит все записи, месяц рождения в которых соответствует введенному с клавиатуры:



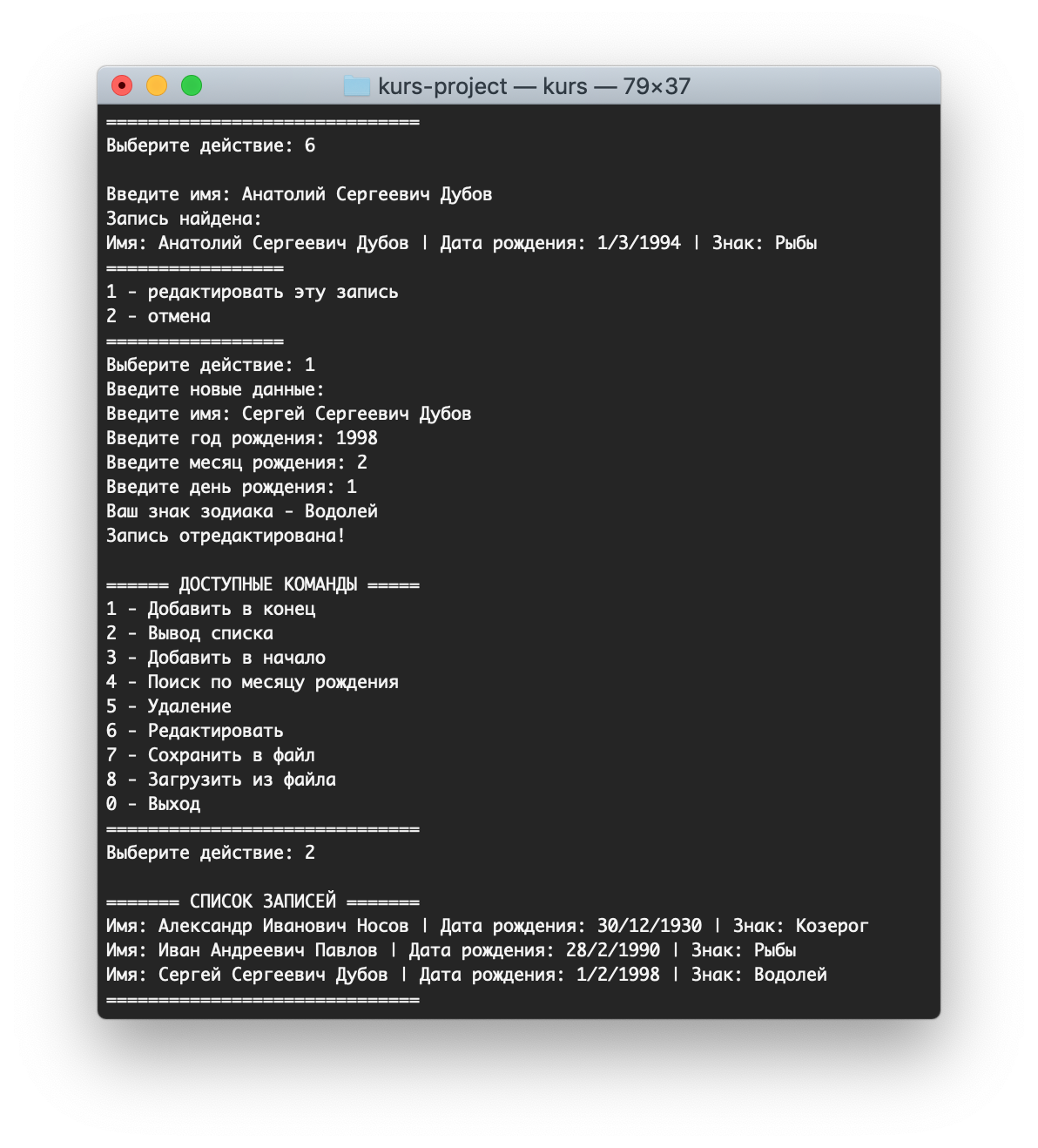
4.5 Удаление

Производит удаление записи:



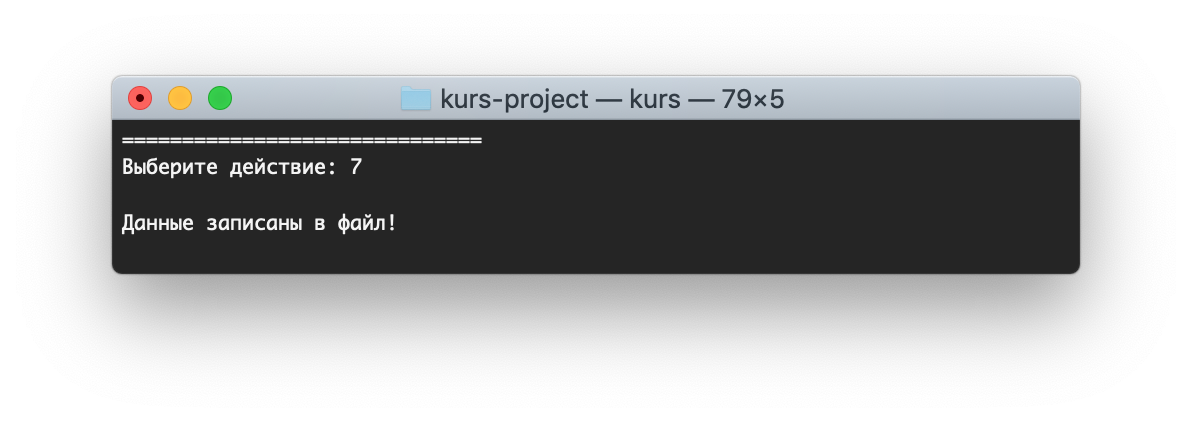
4.6 Редактировать

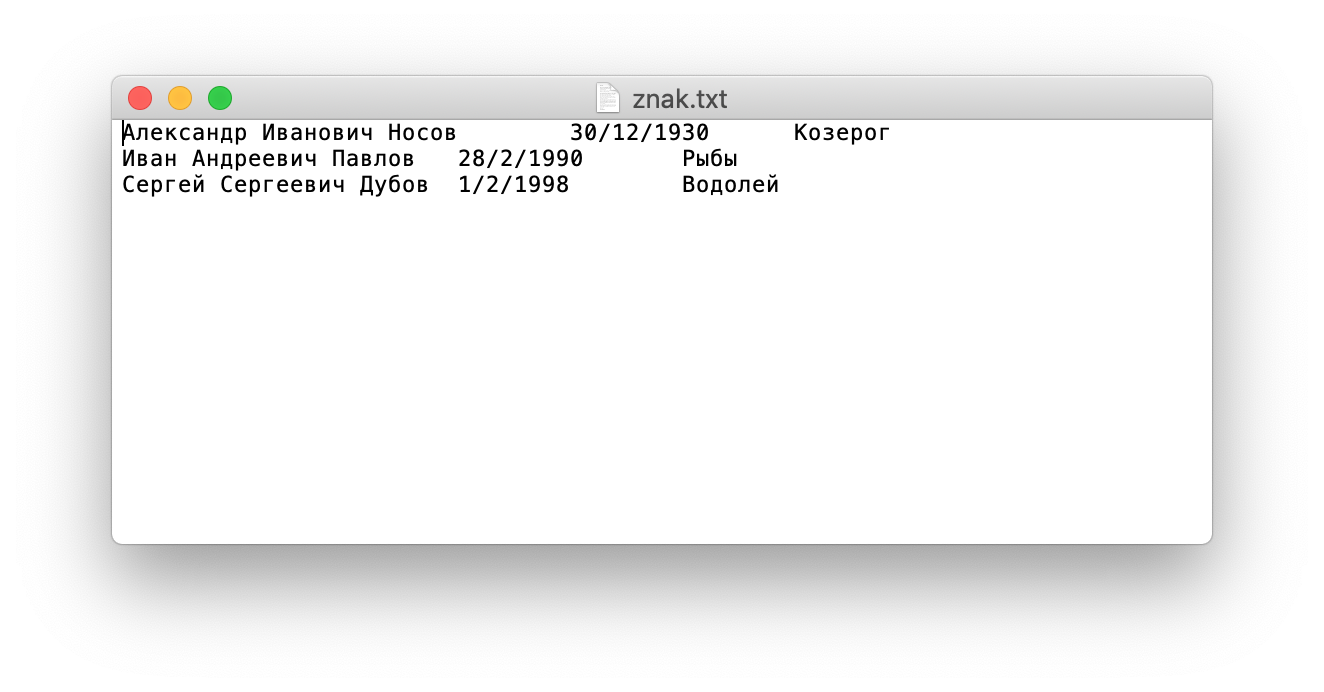
Позволяет редактировать запись:



4.7 Сохранить в файл

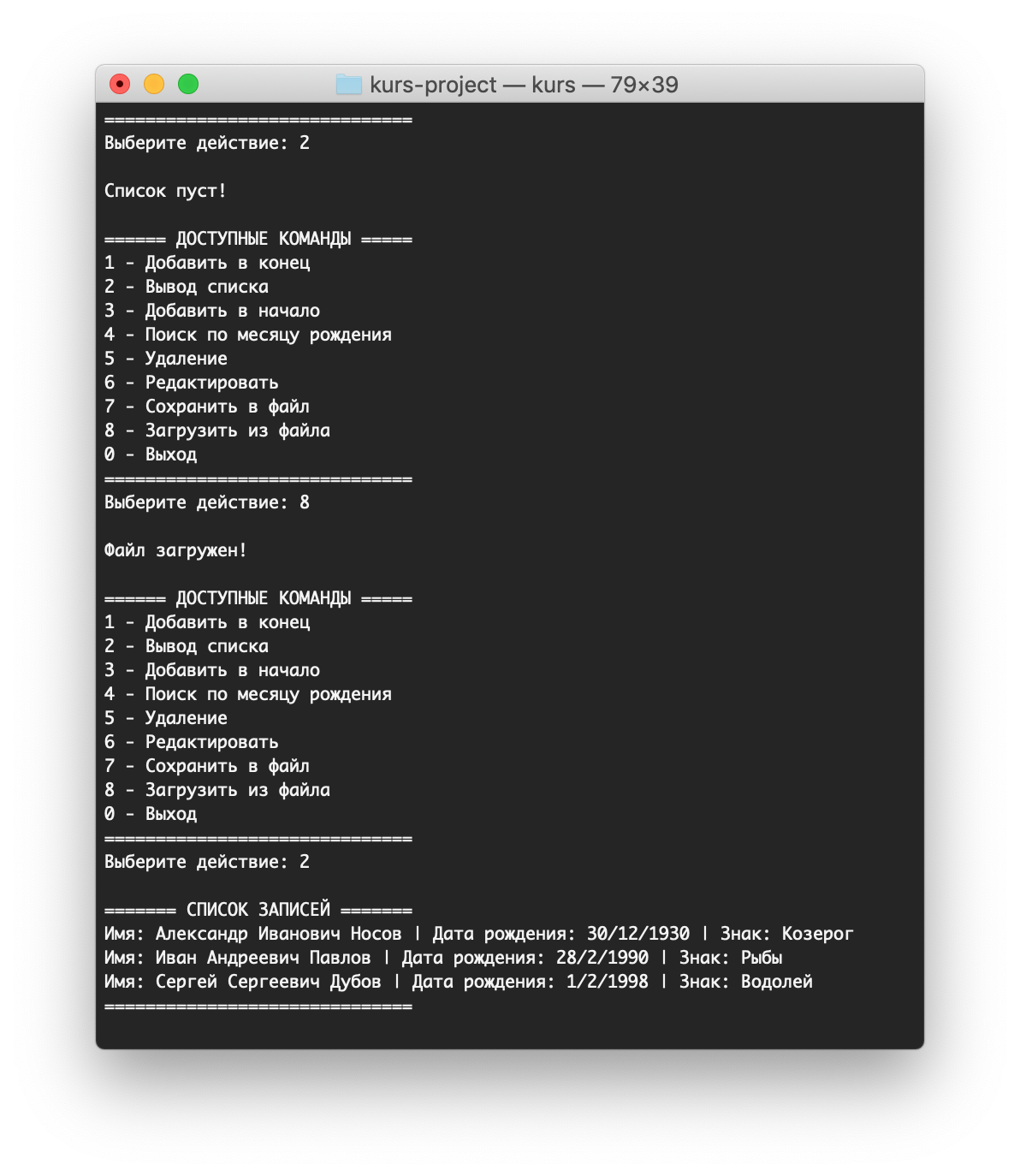
Сохраняет структуру в текстовый файл znak.txt, в соответствии с форматом, указанным в пункте 2.





4.8 Загрузить из файла

Загружает в программу данные из текстового файла znak.txt



**5. Листинг программы**

*// Предметная область – «Информация о сотрудниках».*

*// Данные о человеке хранятся в структуре с именем ZNAK,*

*// содержащей следующие поля:*

*// - фамилия, имя;*

*// - знак зодиака;*

*// - дата рождения (массив из трех чисел).*

*// Задание на поиск: найти информацию о людях,*

*// родившихся в месяц, значение которого*

*// введено с клавиатуры.*

#include <stdlib.h>

#include <iostream>

#include <string>

#include <limits>

#include <cstring>

#include <ctype.h>

#include <iomanip>

#include <time.h>

#include <fstream>

#include <sstream>

**using** **namespace** std;

**const** **int** NAME\_LENGTH = 512;

**const** **int** ZODIAC\_LENGTH = 18;

**const** **int** MIN\_ALLOWED\_BIRTH\_YEAR = 1900;

**struct** ZNAK { *// структура ZNAK*

**char** name[NAME\_LENGTH]; *// имя*

**char** zodiaс[ZODIAC\_LENGTH]; *// знак зодиака*

**int** birthDate[3]; *// дата рождения*

};

**struct** List { *// связный список для хранения записей*

ZNAK data;

List\* next;

};

*// Вывод записи*

**void** printRecord(ZNAK record) {

cout << "Имя: " << record.name << " | "

<< "Дата рождения: " << record.birthDate[0] << "/"

<< record.birthDate[1] << "/" << record.birthDate[2]

<< " | " << "Знак: " << record.zodiaс << endl;

}

*//вывод всех элементов списка на экран*

**void** ShowList(List \*begin) {

List \* p = begin;

cout << "======= СПИСОК ЗАПИСЕЙ =======" << endl;

**while** (p!=**NULL**) {

printRecord(p -> data);

p = p -> next;

}

cout << "==============================" << endl;

}

*// Добавление элемента в конец списка*

**void** AddElem(List \*\*begin, List \*\* cur, ZNAK elem) {

List \* p = **new** List;

p -> data = elem;

*//проверка, является ли список пустым*

**if** (\*begin == **NULL**) {

p -> next = **NULL**;

\*begin = p;

} **else** {

p -> next = (\*cur) -> next;

(\*cur) -> next = p;

}

\*cur = p;

}

*// Добавление элемента в начало списка*

**void** AddFirstElem(List \*\*begin, ZNAK elem) {

List \* p = **new** List;

p -> data = elem;

*//проверка, является ли список пустым*

**if** (\*begin == **NULL**) {

p -> next = **NULL**;

} **else** {

p->next=\*begin;

}

\*begin = p;

}

*// поиск элемента в списке по имени*

List \* FindElem(List \*begin, ZNAK elem) {

List \* p = begin;

**while** (p != **NULL**) {

**if** (!strcmp(p -> data.name, elem.name)) {

**break**;

}

p = p -> next;

}

**return** p;

}

*// поиск элемента в списке по месяцу рождения и их вывод*

**void** FindElem(List \*begin, **int** month) {

List \* p = begin;

**bool** f = **false**;

cout << endl << "====== РЕЗУЛЬТАТЫ ПОИСКА =====" << endl;

**while** (p != **NULL**) {

**if** (p -> data.birthDate[1] == month) {

f = **true**;

printRecord(p -> data);

}

p = p -> next;

}

**if** (!f) {

cout << "Данные не найдены" << endl;

}

cout << "==============================" << endl;

}

*// Удаление элемента из списка*

**void** DelElem(List \*\*begin, List\* ptr, List \*\* cur) {

List \* p;

**if** (ptr == \*begin) { *// удаляем первый элемент*

\*begin = (\*begin) -> next;

} **else** {

*// устанавливаем вспомогательный указатель на элемент,*

*// предшествующий удаляемому*

p =\* begin;

**while** (p -> next != ptr) {

p = p -> next;

}

*// удаление элемента*

p -> next = ptr -> next;

}

*// если это был последний элемент, записать в cur новый*

*// последний элемент*

**if** (ptr -> next == **NULL**) {

\*cur = p;

}

**delete** ptr;

}

*// Очистка памяти*

**void** Free(List \*\*begin) {

**if** (\*begin == 0) **return**;

List \*p = \*begin;

List \*t;

**while** (p) {

t = p;

p = p -> next;

**delete** t;

}

\*begin = **NULL**;

}

*// Определение знака зодиака по дате*

**void** calculateZodiac(ZNAK & record) {

string zodiaс = "";

**switch** (record.birthDate[1]) {

**case** 1:

**if** (record.birthDate[0] < 20)

zodiaс = "Козерог";

**else**

zodiaс = "Водолей";

**break**;

**case** 2:

**if** (record.birthDate[0] < 19)

zodiaс = "Водолей";

**else**

zodiaс = "Рыбы";

**break**;

**case** 3:

**if** (record.birthDate[0] < 21)

zodiaс = "Рыбы";

**else**

zodiaс = "Овен";

**break**;

**case** 4:

**if** (record.birthDate[0] < 20)

zodiaс = "Овен";

**else**

zodiaс = "Телец";

**break**;

**case** 5:

**if** (record.birthDate[0] < 21)

zodiaс = "Телец";

**else**

zodiaс = "Близнецы";

**break**;

**case** 6:

**if** (record.birthDate[0] < 21)

zodiaс = "Близнецы";

**else**

zodiaс = "Рак";

**break**;

**case** 7:

**if** (record.birthDate[0] < 23)

zodiaс = "Рак";

**else**

zodiaс = "Лев";

**break**;

**case** 8:

**if** (record.birthDate[0] < 23)

zodiaс = "Лев";

**else**

zodiaс = "Дева";

**break**;

**case** 9:

**if** (record.birthDate[0] < 23)

zodiaс = "Дева";

**else**

zodiaс = "Весы";

**break**;

**case** 10:

**if** (record.birthDate[0] < 23)

zodiaс = "Весы";

**else**

zodiaс = "Скорпион";

**break**;

**case** 11:

**if** (record.birthDate[0] < 22)

zodiaс = "Скорпион";

**else**

zodiaс = "Стрелец";

**break**;

**case** 12:

**if** (record.birthDate[0] < 22)

zodiaс = "Стрелец";

**else**

zodiaс ="Козерог";

**break**;

}

cout << "Ваш знак зодиака - " << zodiaс << endl;

strcpy(record.zodiaс, zodiaс.c\_str());

}

*// Проверка является ли год високосным*

**bool** leapYear(**int** year) {

**return** (year % 100 != 0 && year % 4 == 0) || (year % 400 == 0);

}

*// Ввод данных*

**void** inputData(List \*begin, ZNAK & record) {

*// если перед cin.getline использовалось cin >> var,*

*// нужно очистить буфер от '\n'*

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

*// Ввод имени*

**bool** isNameCorrect = **false**;

**while** (!isNameCorrect) {

cout << "Введите имя: ";

cin.getline(record.name, NAME\_LENGTH, '\n');

*// если ввели строку длиннее, чем выделено памяти, просто обрезаем*

*// ее и чистим буфер*

**if** (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

*// Проверка уникальности имени*

List\* ptr = FindElem(begin, record);

**if** (strcmp(record.name, "") == 0) {

cout << "Имя не может быть пустым!" << endl;

} **else** **if** ((ptr != **NULL**) && (&(ptr -> data) != &record)) {

cout << "Человек с таким именем уже есть!" << endl;

} **else** {

isNameCorrect = **true**;

}

}

*// Ввод даты рождения*

*// Получить текущее время для проверки корректности ввода*

time\_t theTime = time(**NULL**);

**struct** tm \*currentTime = localtime(&theTime);

**int** day = currentTime -> tm\_mday;

**int** month = currentTime -> tm\_mon + 1; *// Month is 0 - 11*

**int** year = currentTime -> tm\_year + 1900; *// Year is # years since 1900*

*// Воод года от MIN\_ALLOWED\_BIRTH\_YEAR до текущего*

**bool** isBirthDateCorrect = **false**;

**while** (!isBirthDateCorrect) {

cout << "Введите год рождения: ";

cin >> record.birthDate[2];

**if** (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

**if** (

(record.birthDate[2] > year) ||

(record.birthDate[2] < MIN\_ALLOWED\_BIRTH\_YEAR)

) {

cout << "Год рождения введен неправильно!" << endl;

} **else** {

isBirthDateCorrect = **true**;

}

}

*// Ввод месяца - от 1 до 12 или до текущего, если год - текущий*

isBirthDateCorrect = **false**;

**while** (!isBirthDateCorrect) {

cout << "Введите месяц рождения: ";

cin >> record.birthDate[1];

**if** (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

**if** ( ((record.birthDate[2] == year) && (record.birthDate[1] > month))

*// ^ Если год рождения - текущий год, но введенный месяц*

*// еще не наступил*

|| (record.birthDate[1] < 1) || (record.birthDate[1] > 12)

) {

cout << "Месяц рождения введен неправильно!" << endl;

} **else** {

isBirthDateCorrect = **true**;

}

}

*// Ввод дня с проверкой что он не в будещем и что такой номер дня*

*// допустим для введенного месяца, с учетом високосных годов*

isBirthDateCorrect = **false**;

**while** (!isBirthDateCorrect) {

cout << "Введите день рождения: ";

cin >> record.birthDate[0];

**if** (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

**if** ( ((record.birthDate[2] == year) && (record.birthDate[1] == month)

&& (record.birthDate[0] > day))

*// ^ Если год рождения - текущий год, месяц рождения - текущий*

*// месяц, но введенный день еще не наступил*

|| record.birthDate[0] < 1

) {

cout << "День рождения введен неправильно!" << endl;

} **else** {

**switch**(record.birthDate[1]) { *// это месяц*

**case** 2:

*// Високосный год важен только если это Февраль*

**if** (leapYear(record.birthDate[2])) {

**if** (record.birthDate[0] <= 29) {

isBirthDateCorrect = **true**;

}

} **else** {

**if** (record.birthDate[0] <= 28) {

isBirthDateCorrect = **true**;

}

}

**break**;

**case** 1: **case** 3: **case** 5: **case** 7: **case** 8: **case** 10: **case** 12:

**if**(record.birthDate[0] <= 31)

isBirthDateCorrect = **true**;

**break**;

**case** 4: **case** 6: **case** 9: **case** 11:

**if**(record.birthDate[0] <= 30)

isBirthDateCorrect = **true**;

**break**;

}

**if** (!isBirthDateCorrect) {

cout << "День рождения введен неправильно!" << endl;

}

}

}

*// Определение и запись знака зодиака по введенной дате*

calculateZodiac(record);

}

*// Ввод имени для поиска записи*

**void** inputName(ZNAK & record) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Введите имя: ";

cin.getline(record.name, NAME\_LENGTH, '\n');

**if** (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

*// Запись в файл*

**void** saveToFile(List \*begin) {

ofstream file("znak.txt");

**if** (!file) {

cout << "Не удалось открыть файл znak.txt" << endl;

**return**;

}

List \* p = begin;

**while** (p != **NULL**) {

file << (p -> data).name << "\t"

<< (p -> data).birthDate[0] << "/"

<< (p -> data).birthDate[1] << "/"

<< (p -> data).birthDate[2]

<< "\t" << (p -> data).zodiaс << endl;

p = p -> next;

}

cout << "Данные записаны в файл!" << endl;

}

*// Парсинг строки из файла*

**void** readLineFromFile(string& inputLine, List \*begin, ZNAK & record) {

stringstream line(inputLine);

string tempStr;

**short** cnt = 0;

*// Считать из строки разделенные пробелами значения*

**while**(getline(line, tempStr, '\t')) {

*// Считать имя*

**if**(cnt == 0) {

strcpy(record.name, tempStr.c\_str());

}

*// Считать дату*

**else** **if** (cnt == 1) {

stringstream dd(tempStr);

string dateTempStr;

**short** dateCnt = 0;

**while** (getline(dd, dateTempStr, '/')) {

**if** (dateCnt == 0)

record.birthDate[0] = atoi(dateTempStr.c\_str());

**else** **if**(dateCnt == 1) {

record.birthDate[1] = atoi(dateTempStr.c\_str());

} **else** **if**(dateCnt == 2) {

record.birthDate[2] = atoi(dateTempStr.c\_str());

}

++dateCnt;

}

*// Считать знак*

} **else** **if** (cnt == 2) {

strcpy(record.zodiaс, tempStr.c\_str());

}

++cnt;

}

}

*// Чтение из файла*

**void** readFromFile(List \*\*head, List \*begin, List \*\*cur, ZNAK & record) {

ifstream file("znak.txt");

**if** (!file) {

cout << "Не удалось открыть файл znak.txt" << endl;

**return**;

}

string line;

**while** (getline(file, line, '\n')) {

readLineFromFile(line, begin, record);

AddElem(head, cur, record);

}

cout << "Файл загружен!" << endl;

}

**int** main() {

List\* head = **NULL**;

List\* cur = **NULL**;

setlocale(LC\_ALL, "russian");

**int** n = -1;

**int** submenuInput = -1;

ZNAK record;

*// Меню пользователя*

**while** (n != 0) {

cout << endl <<

"====== ДОСТУПНЫЕ КОМАНДЫ =====" << endl <<

"1 - Добавить в конец" << endl <<

"2 - Вывод списка" << endl <<

"3 - Добавить в начало" << endl <<

"4 - Поиск по месяцу рождения" << endl <<

"5 - Удаление" << endl <<

"6 - Редактировать" << endl <<

"7 - Сохранить в файл" << endl <<

"8 - Загрузить из файла" << endl <<

"0 - Выход" << endl <<

"==============================" << endl <<

"Выберите действие: ";

cin >> n;

cout << endl;

**while** (cin.fail() || (n < 0) || (n > 8)) {

cout << "Ошибка ввода, выберите действие: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> n;

}

**switch** (n) {

**case** 1: {

inputData(head, record);

AddElem(&head, &cur, record);

cout << "Запись добавлена!" << endl;

**break**;

} **case** 2: {

**if** (head) {

ShowList(head);

} **else** {

cout << "Список пуст!" << endl;

}

**break**;

} **case** 3: {

inputData(head, record);

AddFirstElem(&head, record);

cout << "Запись добавлена!" << endl;

**break**;

} **case** 4: {

**if** (!head) {

cout << "Список пуст!" << endl;

**break**;

}

**int** month;

cout << "Введите месяц: ";

cin >> month;

FindElem(head, month);

**break**;

} **case** 5: {

**if** (!head) {

cout << "Список пуст!" << endl;

**break**;

}

List\* ptr;

inputName(record);

ptr = FindElem(head, record);

**if** (ptr == **NULL**) {

cout << "Запись не найдена!" << endl;

} **else** {

DelElem(&head, ptr, &cur);

cout << "Запись удалена!" << endl;

}

**break**;

} **case** 6: {

**if** (!head) {

cout << "Список пуст!" << endl;

**break**;

}

List\* ptr;

inputName(record);

ptr = FindElem(head, record);

**if** (ptr == **NULL**) {

cout << "Запись не найдена!" << endl;

} **else** {

cout << "Запись найдена: " << endl;

printRecord(ptr -> data);

cout << "=================" << endl;

cout << "1 - редактировать эту запись" << endl;

cout << "2 - отмена" << endl;

cout << "=================" << endl;

cout << "Выберите действие: ";

cin >> submenuInput;

**while** (cin.fail() || (submenuInput < 1) || (submenuInput > 2)) {

cout << "Ошибка ввода, выберите действие: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> submenuInput;

}

**if** (submenuInput == 2) **break**;

cout << "Введите новые данные: " << endl;

inputData(head, ptr -> data);

cout << "Запись отредактирована!" << endl;

}

**break**;

} **case** 7: {

**if** (head) {

saveToFile(head);

} **else** {

cout << "Список пуст!" << endl;

}

**break**;

} **case** 8: {

Free(&head);

readFromFile(&head, head, &cur, record);

}

}

}

Free(&head);

**return** 0;

}

**6. Заключение**

Была разработана и протестирована программа для работы с базой данных в текстовом файле. При разработке были изучены или закреплены такие навыки как: работа со структурами данных, например связный список, работа со строками, валидация введенных пользователем данных, запись и чтение файлов, поиск и удаление элементов связного списка.