**АННОТАЦИЯ**

В данной пояснительной записке к курсовой работе описывается принцип работы программы для расчета количества просроченного товара, входные данные задаются в табличном виде.

В данной записке приведены:

* прототипы функций;
* структурные схемы функций;
* содержание файлов (входных, выходных, программы);
* пошаговый разбор функционала программы.

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 4](#_Toc26385424)

[1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ 6](#_Toc26385425)

[2 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 7](#_Toc26385426)

[2.1 Постановка задачи 7](#_Toc26385427)

[2.2 Выбор языка программирования и среды разработки 7](#_Toc26385428)

[2.3 Входные и выходные данные 8](#_Toc26385429)

[2.4 Применяемые математические методы 8](#_Toc26385430)

[2.5 Структура программы 8](#_Toc26385431)

[2.6 Используемые функции 9](#_Toc26385432)

[2.7 Структурные схемы алгоритмов 11](#_Toc26385433)

[3 ВЫПОЛНЕНИЕ ПРОГРАММЫ 24](#_Toc26385434)

[3.1 Условия выполнения программы 24](#_Toc26385435)

[3.2 Запуск программы 24](#_Toc26385436)

[3.3 Проверка работоспособности программы 25](#_Toc26385437)

[**ЗАКЛЮЧЕНИЕ** 29](#_Toc26385438)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc26385439)

[ПРИЛОЖЕНИЕ А. Листинг программы 31](#_Toc26385440)

**ВВЕДЕНИЕ**

Целью данной курсовой работы является закрепление и систематизация полученных знаний в области основ программирования и совершенствовании приобретенных навыков разработки приложений на языке Си с использованием методологии структурного программирования. Данный опыт позволит закрепить опыт полученный в курсе программирования линейных алгоритмов и сформирует хороший фундамент для успешного освоения последующих дисциплин разработки интерактивных программ.

В задачи данного курсового проекта входит:

* детализация постановки задачи;
* выбор типов и проектирования структур данных, определяющих способы хранения, обработки и преобразования входных данных;
* разработка структуры программы;
* написание непосредственно самой программы (кодирование);
* разработка тестовых примеров для корректной отладки и проверки работоспособности всех ветвей программы;
* формирования программной документации в соответствии с действующими стандартами.

В первом разделе данной записки описывается общее назначение и область применения программы.

Второй раздел включает технические характеристики программы: постановка задачи; выбор языка программирования и среды разработки; входные и выходные данные; применяемые в ходе разработке математические методы; структура программы; используемые в ходе разработки функции; структурные схемы основных алгоритмов

Третий раздел содержит описание пошагового тестирования, и демонстрация корректной работы программы с тестовыми примерами включает подпункты: условия выполнения программы; запуск программы; проверка работоспособности программы.

# 1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ

Программа предназначена для обработки табличных данных, содержащих сведения о спортсменах, а также для выполнения их поиска по текущему списку необходимых для редактирования или удаления. Также в функционал программы входит нахождение самых тяжёлых спортсменов и самых старших спортсменок.

Область программы не ограничена она может применяться как для подсчёта количества спортменов, так и для распределения их по командам и для сравнения их возраста и веса.

Программа написана для операционной системы Windows.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

## 2.1 Постановка задачи

Данные входного фала имеет следующий вид:

– ФИО;

– Дата рождения;

– Пол;

– Вес;

– Спортивный разряд;

– Название команды;

Найти самых старших девушек и самых тяжёлых юношей в каждой команде.

## Выбор языка программирования и среды разработки

Программа написана на языке программирования Си с элементами языка Си++ в среде разработки Visual Studio. Данная среда разработки была выбрана, так-как является современной средой разработки и является ПО разработанным крупной компанией в сфере IT-индустрии Microsoft. Данная программа кроссплатформенная и позволяет писать программы на большом количестве языков программирования, также содержит функциональные дополнения для комфортной разработки приложений.

Язык программирования Си был выбран так как является популярным языком программирования. Также данный язык является основой для большинства современных программ.

## Входные и выходные данные

Для успешного запуска и корректной работы программы не стоит разделять файлы, которые находятся непосредственно с самим файлом запуска Kurs.exe

Отсутствие данных файлов с большой вероятностью нарушит работоспособность программы.

В программе предусмотрен функционал для работы как с бинарными, так и текстовыми файлами. При заполнении поля «Дата рождения» не стоит разделять цифры точками, следует вводить число, месяц и год раздельно, т.е. после ввода каждого значения нажимать клавишу Enter. В противном случае запись не будет произведена. Также после завершения работы с таблицей программа автоматически создаёт бинарный файл stdBin.dat.

## Применяемые математические методы

В программе отсутствуют сложные математические методы. Для вычисления веса и роста применялось простое сравнение входящих данных.

## Структура программы

Записи в программе имеют двусвязную циклическую структуру.

Структура, содержащая дату в формате день/месяц/ год:

struct data {

int day, month, year;

};

Структура, содержащая основную информацию о спортсмене:

struct info {

char fio[30];

struct data birthday;

char gender;

float weight;

char rank[30];

char team[30];

};

Структура для реализации двухсвязного циклического списка:

struct spisok {

info inf;

struct spisok\* prev, \* next;

};

Структура для реализации заполнения данных о спортсмене

struct info fillInfo() {

info temp;

cout << "Введите ФИО спортсмена: ";

cin >> temp.fio;

while (cin.peek() != '\n') {

char tmp[15];

cin >> tmp;

strcat(temp.fio, " ");

strcat(temp.fio, tmp);

}

>> temp.weight;

cout << "Введите спортивный разряд: ";

cin

cout << "Введите дату рождения: ";

cin >> temp.birthday.day >> temp.birthday.month >> temp.birthday.year;

cout << "Введите пол [М | Ж]: ";

cin >> temp.gender;

cout << "Введите вес: ";

cin >> temp.rank;

while (cin.peek() != '\n') {

char tmp[15];

cin >> tmp;

strcat(temp.rank, " ");

strcat(temp.rank, tmp);

}

cout << "Введите название команды: ";

cin >> temp.team;

while (cin.peek() != '\n') {

char tmp[15];

cin >> tmp;

strcat(temp.team, " ");

strcat(temp.team, tmp);

}

return temp;

}

## Используемые функции

// Ручное заполнение информационных полей списка

info fillInfo();

// Функия для заполнения информационной структуры из строчки

info createInfoStructFromLine(string line);

// Функция добавления элемента в список

void addItem(spisok\*& Top, spisok\*& Bottom, info \_info, int& countItems);

// Вывод списка на экран с возможностью скрола

void printList(spisok\* Top, int& countItems);

// Вывод текущего элемента в списке

void printCurentLine(spisok\* cur);

// Запись списка в бинарный файл

void writeBinFile(spisok\* Top, string path, int& countItems)

// Создание списка из бинарного файла

void readBinFile(spisok\*& Top, spisok\*& Bottom, string path, int& countItems)

// Удаление элемента

void delItem(spisok\*& Top, spisok\*& Bottom, int& countItems, int pos);

// Вывод на экран пунктов меню

void printMenu(int& pos);

// Поиск самого тяжёлого юноши и самой старшей девушки

void findGirlsAndBoys(spisok\* Top, int& countItems)

// Поиск по ФИО

void findFio(spisok\* Top, char \_fio[30], int& countItems)

// Сортировка по командам

void sortGroup(spisok\* Top, spisok\* Bottom, int& countItems)

// Замена содержимого записи

void replaceInfo(spisok\* Top, int index, int& countItems)

// Создание списка из текстового файла

void readTxtFile(spisok\*& Top, spisok\*& Bottom, string path, int& countItems)

// Запись в текстовый файл

void writeTxtFile(spisok\* Top, string path, int& countItems)

// Добавление элемента в список

void addItem(spisok\*& Top, spisok\*& Bottom, info \_info, int& countItems)

## Структурные схемы алгоритмов

Структурная схема основной функции main(), представлена на рисунке 2.1.

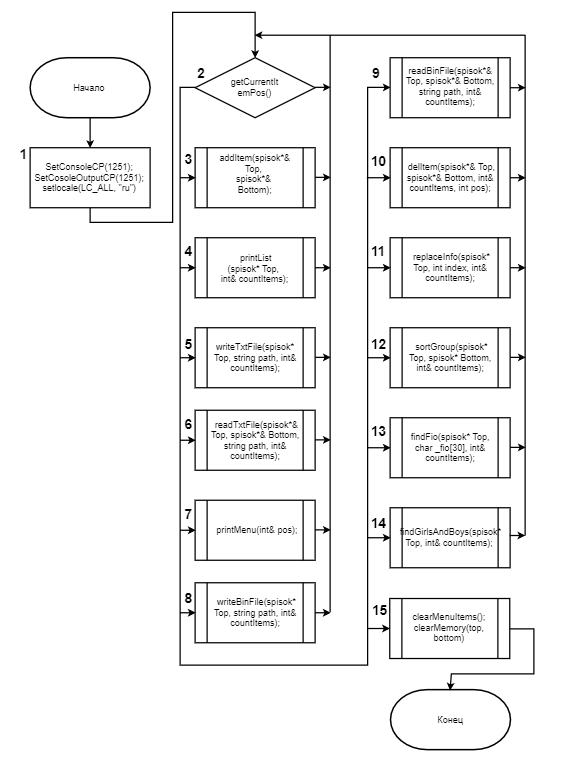


Рисунок 2.1 – Алгоритм основной функции

Функционал блоков алгоритма структурной схемы функции main ():

Блок 1. Установка локализации программы для корректного отображения русского языка на разных функциональных машинах.

Блок 2 (switch). Данный блок отвечает за вызов функций основываясь на текущей позиции основного меню программы.

Блок 3. Вызов функции добавления нового элемента.

Блок 4. Вызов функции создания списка

Блок 5. Вызов функции записи в текстовый файл.

Блок 6. Вызов функции чтения из тестового файла.

Блок 7. Вывод меню на экран пользователя.

Блок 8. Вызов функции сохранения текущего списка бинарный файл.

Блок 9. Вызов функции чтения из бинарного файла.

Блок 10. Вызов функции удаления из спика.

Блок 11. Вызов функции редактирования записи

Блок 12. Вызов фунции сортировки по командам.

Блок 13. Вызов функции поиска по ФИО.

Блок 14. Вызов функции поиска старших девушек и тяжёлых парней в каждой команде.

Блок 15. Служит для очистки всей динамической памяти занимаемой программой (очистка списка элементов, очистка памяти для пунктов меню), а также закрытия приложения, срабатывает при выбранной позиции главного меню равном 20.

Алгоритм функции вывода списка на экран, представлена на рисунке 2.2.

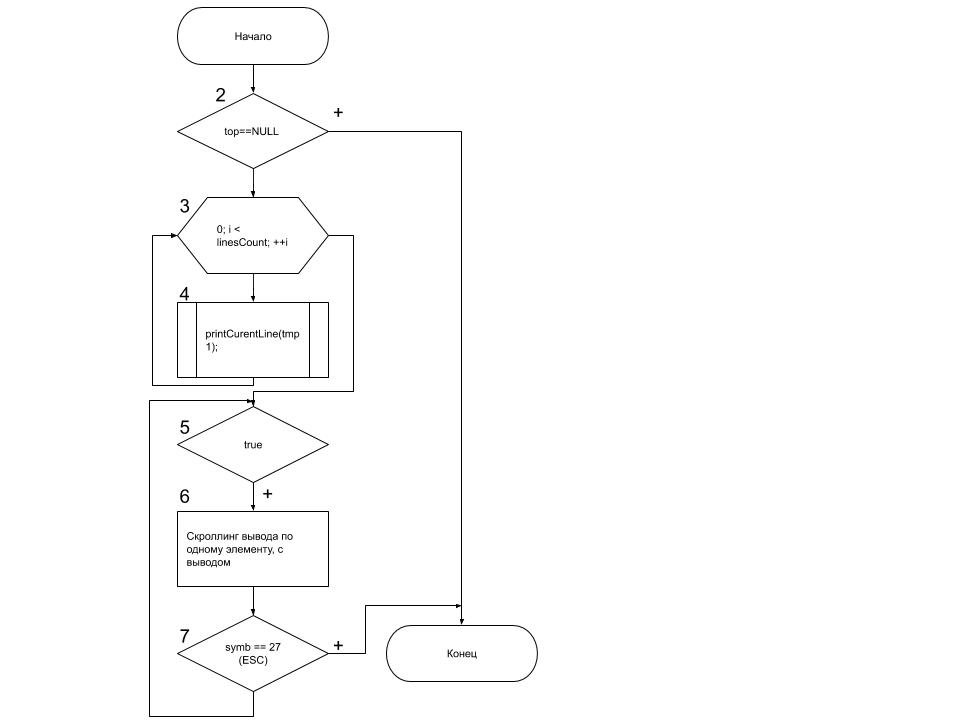


Рисунок 2.2 – Алгоритм вывода списка

Функционал блоков структурной схемы:

Блок 2. Проверка на наличие записей в списке.

Блок 3. Цикл который осуществляет переход по элементам списка.

Блок 4. Выводит текущую запись на итерации цикла.

Блок 5. Бесконечный цикл для реализации скроллинга вывода.

Блок 6. Вызов функций скроллинга в зависимости от нажатой клавиши.

Блок 7. При нажатии клавиши ESC будет выполнен выход из бесконечного цикла, что приведет к завершению функции.

Алгоритм функции добавления записи, представлена на рисунке 2.3.

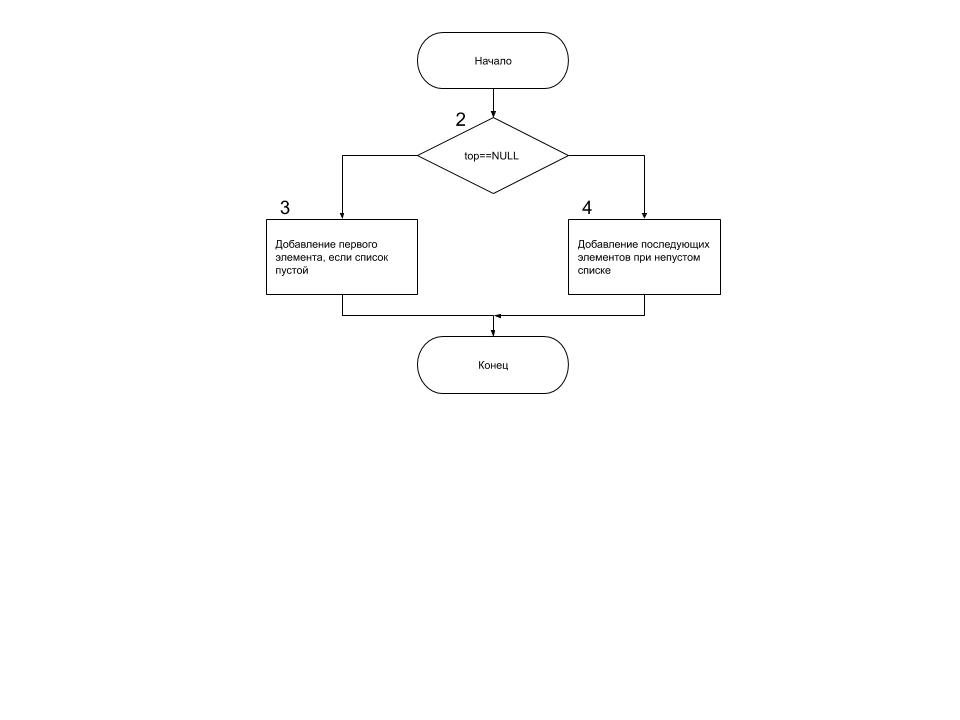


Рисунок 2.3 – Алгоритм добавления записи

Функционал блоков структурной схемы (рисунок 2.3)

Блок 2. Осуществляется проверка на наличие элементов в списке, при значении блока false осуществляется переход к блоку 3.

Блок 3. Добавление записи в начало списка и циклическое связывание блока.

Блок 4. В любом другом случае будет срабатывать блок добавление записи в конец списка.

Алгоритм функции очистки памяти приведен на рисунке 2.4.

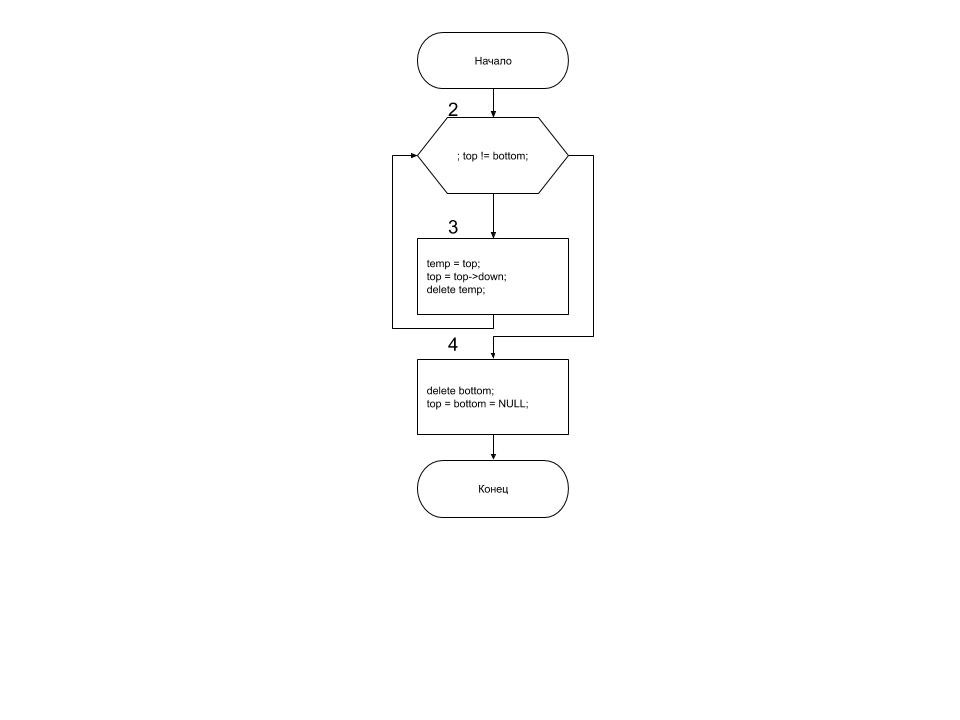


Рисунок 2.4 – Алгоритм очистки памяти

Функционал блоков структурной схемы:

Блок 2. Перебор элементов от верхушки списка до низу.

Блок 3. Удаление текущего элемента и присвоение верхушки следующему.

Блок 4. Так как список циклический Bottom не будет равен NULL, удаляем конечную запись и обнуляем указатели.

Алгоритм поиска по ФИО приведен на рисунке 2.5.

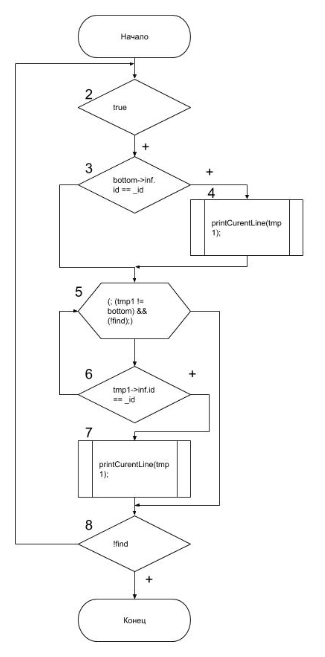


Рисунок – Алгоритм поиска по ФИО

Функционал блоков структурной схемы.

Блок 2. Бесконечный цикл

Блок 3. Проверка последней записи на равенство поля ФИО с указанным параметром.

Блок 4. Если условие Блока 3 истинно, то выводит данную запись.

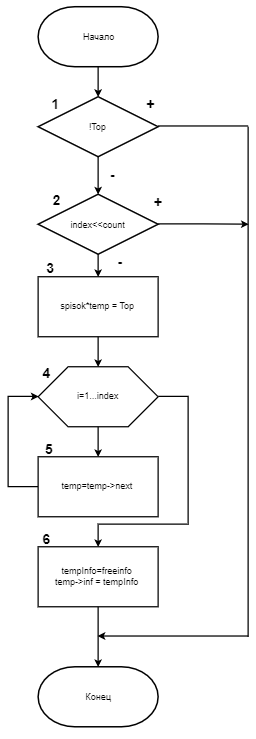
Блок 5. Если условие Блока 3 ложно и запись не найдена, то продолжается поиск до последней записи.

Блок 6. Проверка на равенство полей ФИО текущей записи и параметра.

Блок 7. Вывод текущей записи.

Блок 8. Если не найдено, то повторить.

Алгоритм редактирования записи приведён на рисунке 2.6



Функционал блоков структурной схемы.

Блок 1. Проверка на наличие списка.

Блок 2. Проверка наличия введённого номера в списке.

Блок 3. Инициализация переменной.

Блок 4. Цикл прохода по списку.

Блок 5. Переход на новый элемент.

Блок 6. Очистка данных и замена их новыми значениями.

Алгоритм создания из бинарного файла приведен на рисунке 2.7.

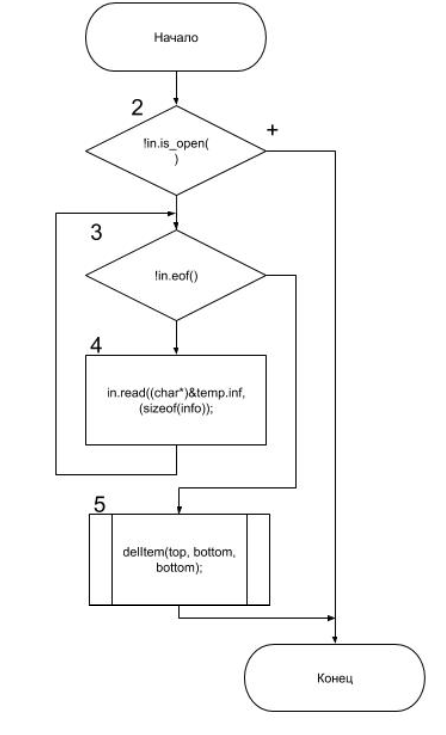


Рисунок 2.7 – Алгоритм создания списка из бинарного файла

Функционал блоков структурной схемы.

Блок 2. Проверка открыт ли файл.

Блок 3. Перебор списка по одному элементу до предпоследнего.

Блок 4. Чтение информационной структуры из файла.

Блок 5. Удаление дублирующего элемента.

Алгоритм создания из текстового файла, приведен на рисунке 2.8.

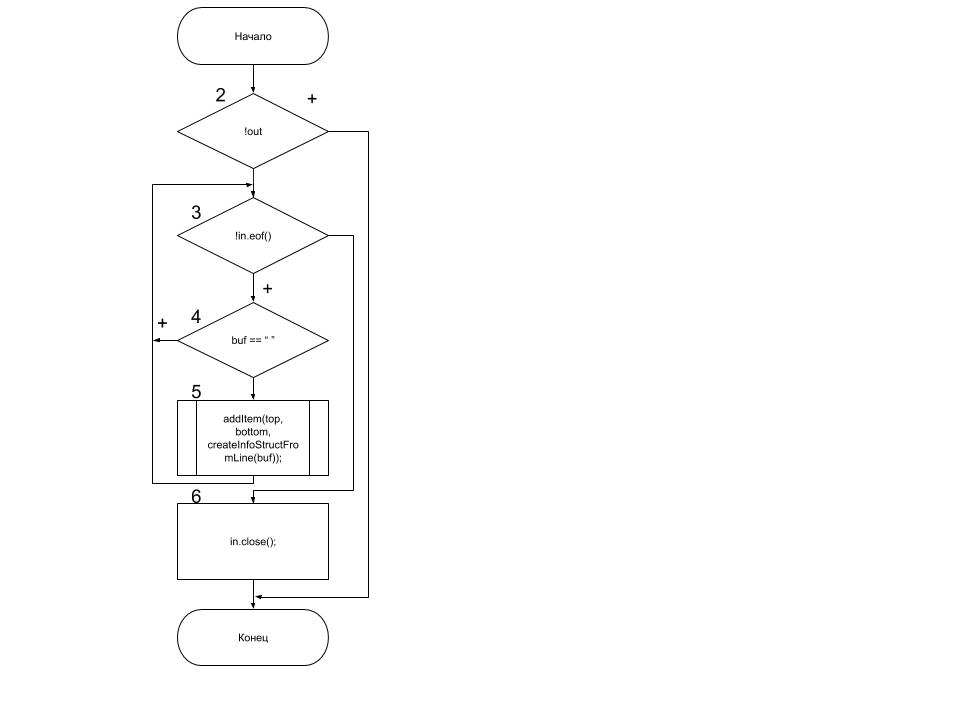


Рисунок 2.8 – Алгоритм создания списка из текстового файла

Функционал блоков структурной схемы.

Блок 2. Проверка, успешно открыт ли файл.

Блок 3. Проверка на конец файла.

Блок 4. Проверка буфера для чтения на наличие символов.

Блок 5. Добавление записи в список.

Блок 6. Закрытие потока чтения.

Алгоритм сортировки по названию (рисунок 2.9).

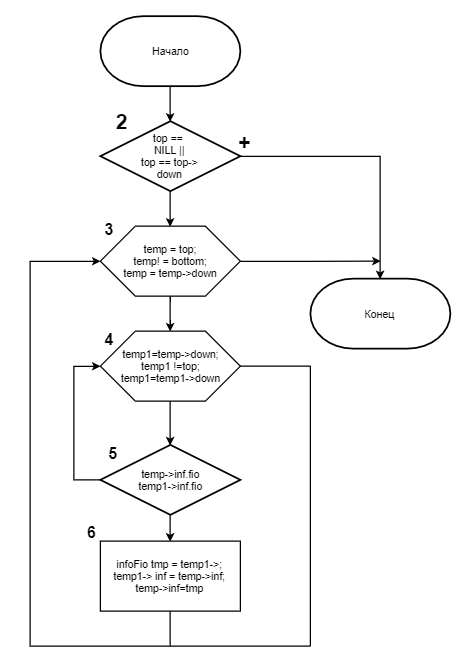


Рисунок 2.9 – Алгоритм сортировки по ФИО

Функционал блоков структурной схемы.

Блок 2. Проверка на наличие в списке элементов и на их количество больше 1.

Блок 3. Перебор элементов по одному до предпоследнего элемента.

Блок 4. Перебор элементов по одному от следующего за элементом выбранном на блоке 3 до конца списка.

Блок 5. Проверка на лексикографическую разницу полей названия.

Блок 6. Перестановка местами содержимого информационных полей.

Алгоритм сохранения в бинарный файл (рисунок 2.10).

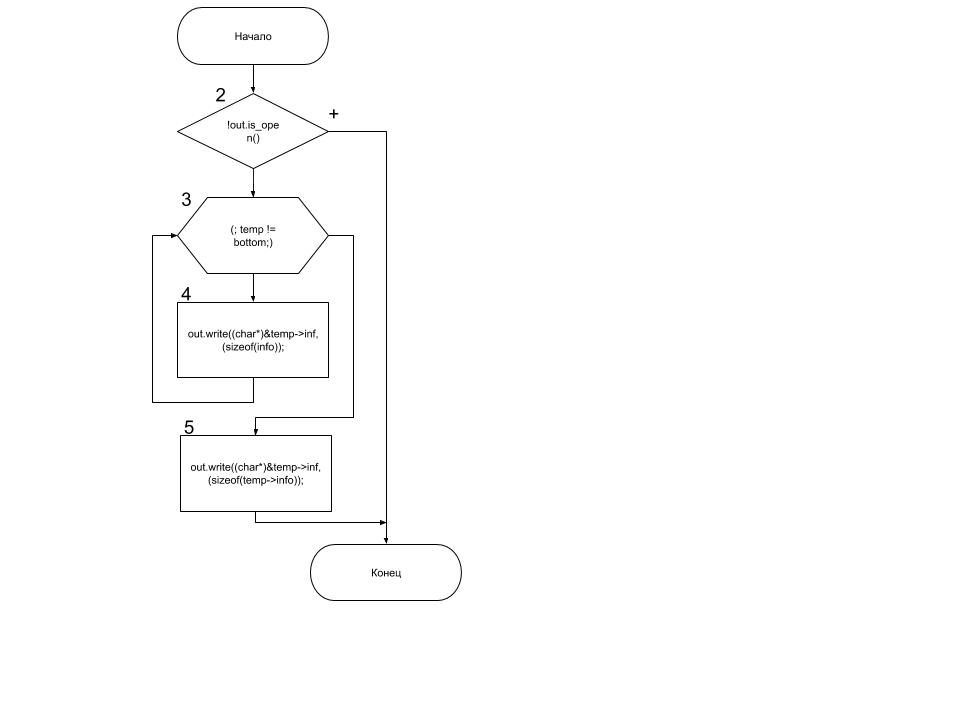


Рисунок 2.10 – Алгоритм сохранения в бинарный файл

Функционал блоков структурной схемы

Блок 2. Проверка, открыт ли файл.

Блок 3. Перебор элементов по одному до предпоследнего элемента.

Блок 4. Запись информационной структуры в файл.

Блок 5. Запись в файл последнего элемента списка.

Алгоритм сохранения в текстовый файл (рисунок 2.11).

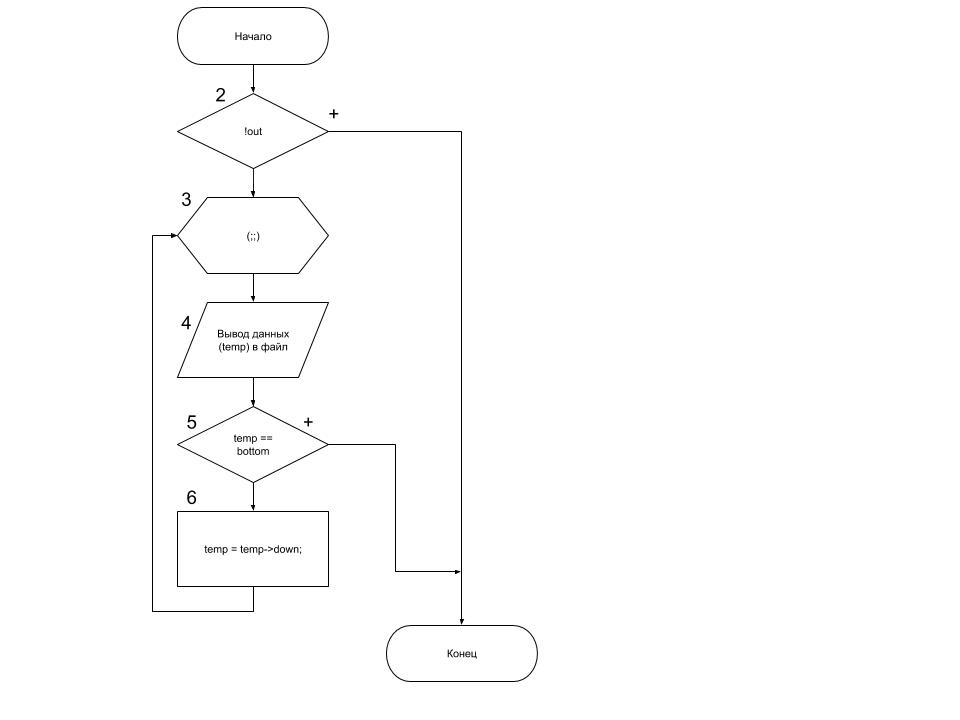


Рисунок 2.11 – Алгоритм сохранения в текстовый файл

Функционал блоков структурной схемы

Блок 2. Проверка открыт ли файл.

Блок 3. Бесконечный цикл.

Блок 4. Вывод построчно форматированного текста в файл.

Блок 5. Проверка, равен ли текущий элемент конечному элементу.

Блок 6. Переход на следующий.

Алгоритм удаления записи (рисунок 2.12).

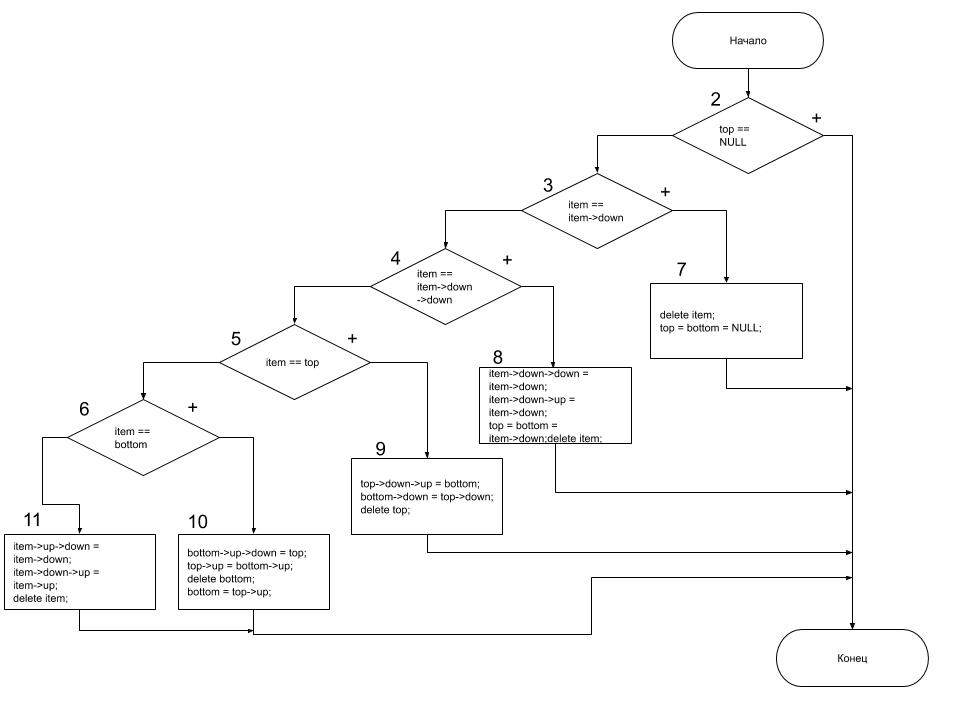


Рисунок 2.12 – Алгоритм удаления элемента из списка

Функционал блоков структурной схемы

Блок 2. Проверка на наличие элементов в списке.

Блок 3. Проверка на наличие одного элемента в списке.

Блок 4. Проверка на наличие 2 элементов списка.

Блок 5. Проверка является ли элемент верхушкой списка.

Блок 6. Проверка является ли элемент последним в списке.

Блок 7. Удаление если в списке один элемент.

Блок 8. Удаление если в списке два элемента.

Блок 9. Удаление первого элемента из списка.

Блок 10. Удаление нижнего элемента из списка.

Блок 11. Удаление в любом другом случае.

# 3 ВЫПОЛНЕНИЕ ПРОГРАММЫ

## Условия выполнения программы

Компиляция программы произведена под 32 – разрядную ОС Windows.

* Корректность работы программы проверена на ОС – Windows 7, Windows 8.1, Windows 10, при запуске программы возможно некорректное отображение русского текста, связанно это с стандартным шрифтом консольного приложения, для правильного отображения текста рекомендуется поставить в настройках консоли текст Consolas. Скомпилированная программа состоит всего из одного файла Kurs.exe. При первом запуске программы и завершении работы с ней создаётся бинарный файл stdBin.dat (файл авто сохранения в бинарный файл).

Данная программа не требовательная к системе и будет корректно работать на любом компьютере.

## Запуск программы

Для запуска программы необходимо запустить файл Kurs.exe

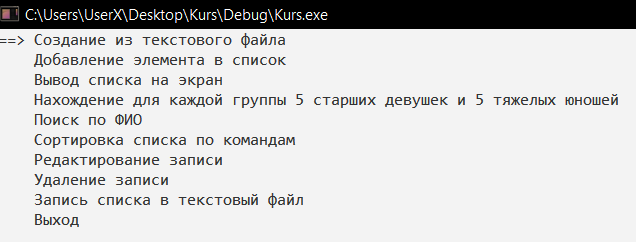


Рисунок 3.1 – Запуск программы

При запуске программы появляется консоль с пунктами меню, перемещение между пунктами меню осуществляется стрелочками (вверх и вниз), выбор выделенного пункта меню однократным нажатием клавиши Enter. С помощью данного меню пользователь имеет возможность создать новый список из файла или начать ввод вручную, при этом в дальнейшем можно объединить несколько списков в один и редактировать их. Также имеется возможность выполнить сортировку спортсменов по командам, найти самых крупных и самых старших девушек в каждой из команд и поиска участников соревнований по ФИО. При завершении работы с программой она автоматически сохраняет результат работы как в бинарный, так и в текстовый файл, которые потом можно открыть для продолжения работы.

## 3.3 Проверка работоспособности программы

При первом запуске программы не будет никаких записей и нужно выбрать способ создания базы данных. Для демонстрации был выбран текстовый файл с содержимым изображенным на (рисунок 3.2).

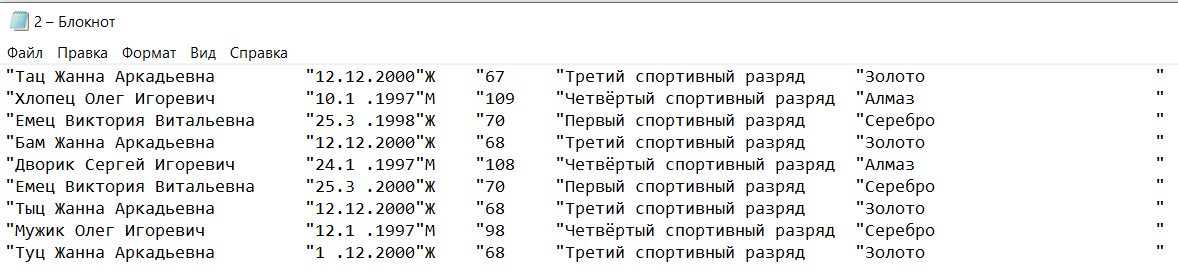


Рисунок 3.2 – Содержимое ранее сформированного файла

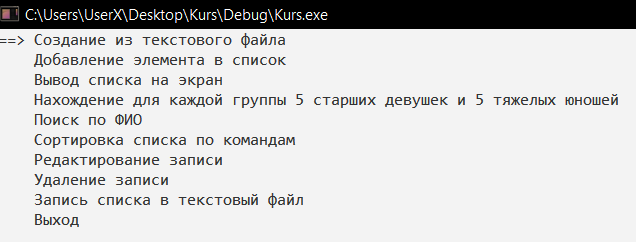


Рисунок 3.3 – Меню программы

Выберем пункт вывода списка, после выбора данного пункта меню необходимо ввести название файла из которого будет реализован список (рисунок 3.3)

E:\Схемы курсач\Скрины\Создание списка из файла.png

Рисунок 3.3 – Создание списка из файла

Скроллинг записей осуществляется стрелочками (вверх и вниз). Также можно выполнить редактирование записи, выбрав соответсвующий пункт меню (рисунок 3.4)

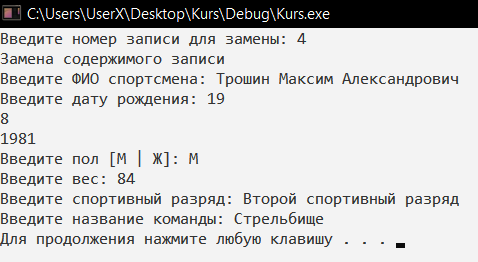


Рисунок 3.4 – Вывод списка на экран

При успешном редактировании записи в списке, на месте её исходных данных, появятся изменённые, введённые пользователем (рисунок 3.5)

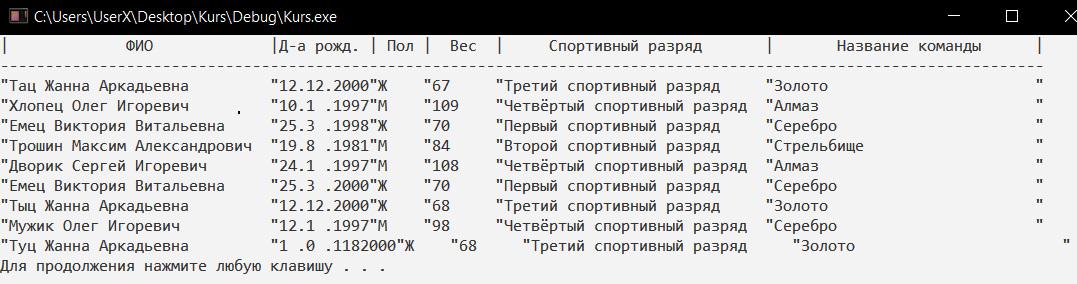


Рисунок 3.5 –Вид списка после редактирования записи

В программе реализован алгоритм поиска записи по полю ФИО. В программе установлено ограничения в поле ФИО (допустимый диапазон ввода до 29 символов),

При попытке ввода программа проверяет корректность ввода и при успешном поиске выводит всех людей с введенным ФИО, в противном случае ничего не будет выведено на экран (рисунок 3.6)

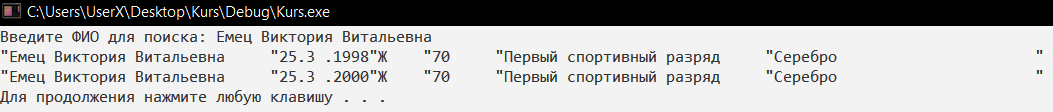


Рисунок 3.6 – Поиск спортсмена по ФИО

В программе реализован алгоритм поиска самых старших девушек и самых крупных юношей в каждой из команд. При выборе пункта меню «Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей» на экран будет выведен список парней и девушек, удовлетворяющих условию, распределённых по командам. Если в команде не присутствют лица мужского или женского пола, будет выведено соответсвующее сообщение (рисунок 3.7)

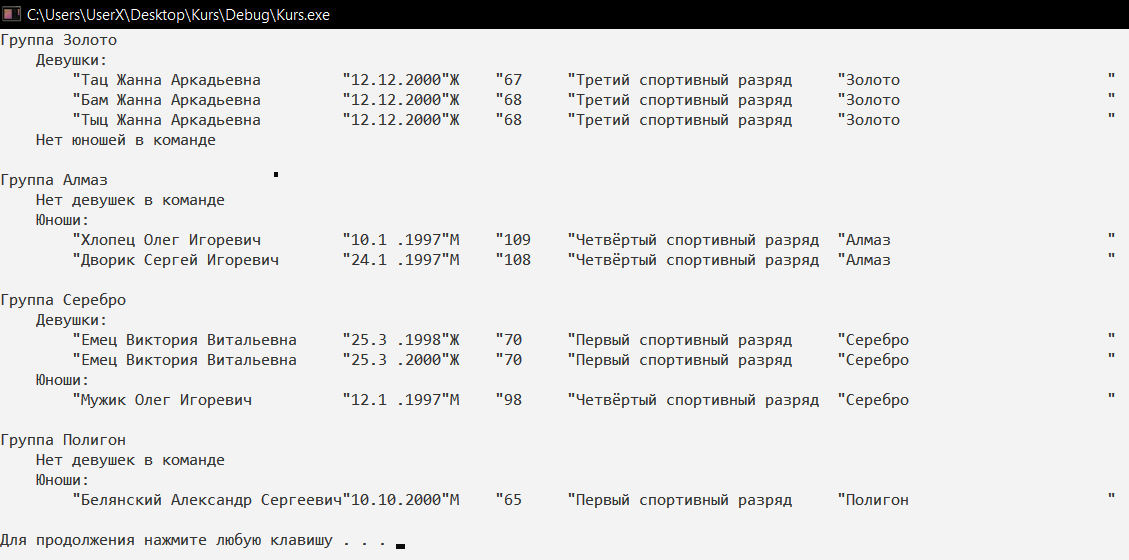


Рисунок 3.7 – Сообщение об некорректном вводе поля Name

В программе реализованы функции записи списка в бинарный файл и текстовый. Запись в текстовый файл осуществляется при выборе соответствующих пунктах меню. После выбора пункта нужно ввести путь для сохранения, при этом нужно ввести расширение файла. Затем его можно будет просматривать с помощью приложения Блокнот. Запись в бинарный файл осуществляется автоматически при выходе из программы.

Функция сохранения в бинарный файл вызывается автоматически при корректном завершении работы с программой и выборе соответствующего пункта меню Выход.

В программе реализована сортировка участника по командам. Для этого нужно выбрать соответсвующий пункт меню, после чего сортировка будет произведена (рисунок 3.8)

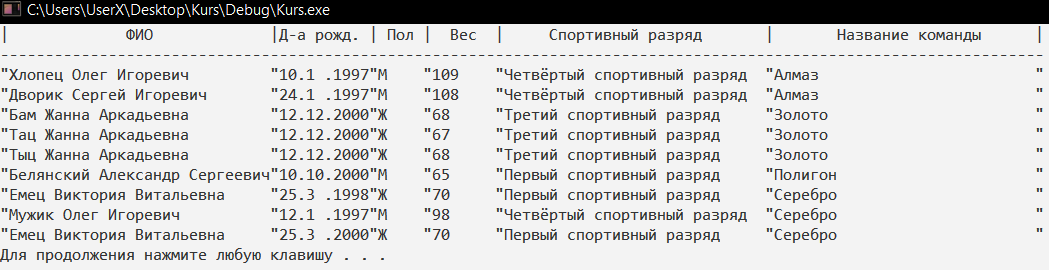


Рисунок 3.8 – Вид списка после сортировки

Удаление записи производится через пункт меню «Удаление записи». После выбора данного пункта пользователю стоит ввести номер записи, которую он хочет удалить. При успешном удалении записи, пункт списка, номер которого ранее введён пользователем, будет исключён из списка, в противном случае на экран будет выведено сообщение о том, что пункта под таким номером не сущетсвует (рисунок 3.9)

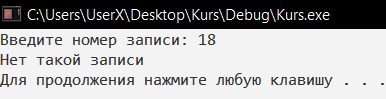


Рисунок 3.9 –Удаление записи

Для завершения работы с программой следует выбрать пункт меню Выход, чтобы корректно завершить выполнение приложения и избежать потери новых данных.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы была достигнута поставленная цель, а именно: систематизация и совершенствование практических навыков разработки программ на языке Си с использованием методологии структурного программирования. На базе полученных навыков программирования была написана программа для управления базой данных, также в ходе написания программы были изучены принципы построения двусвязных циклических списков.

Реализован весь функционал, который требуется в соответствии с общим заданием, а также алгоритм, который необходим по индивидуальному заданию (поиск самого тяжёлого юноши и самой высокой девушки).

Программа протестирована на различных сценариях, которые затрагивали каждую ветвь программы, также производилась проверка на корректность ввода данных.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Павловская Т.А. C/C++, Структурное программирование, практикум/ Павловская Т.А., Щупак Ю.А. – Санкт-Петербург. Изд-во «Питер», 2003 – 238 с.

2) Справочник по Windows API [Электронный ресурс]. – URL: http://www.vsokovikov.narod.ru/New\_MSDN\_API/ref\_api.htm (дата обращения: 10.11.2019).

# ПРИЛОЖЕНИЕ А.

# Листинг программы

**Файл Source.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string.h>

#include <stdlib.h>

#include <fstream>

#include <iomanip>

#include <conio.h>

#include <locale.h>

#include <Windows.h>

using namespace std;

struct data {

int day, month, year;

};

struct info {

char fio[30];

struct data birthday;

char gender;

float weight;

char rank[30];

char team[30];

};

struct spisok {

info inf;

struct spisok\* prev, \* next;

};

struct info fillInfo();

void addItem(spisok\*& Top, spisok\*& Bottom, info \_info, int& countItems);

void printList(spisok\* Top, int& countItems);

void writeTxtFile(spisok\* Top, string path, int& countItems);

void readTxtFile(spisok\*& Top, spisok\*& Bottom, string path, int& countItems);

void printMenu(int& pos);

void writeBinFile(spisok\* Top, string path, int& countItems);

void readBinFile(spisok\*& Top, spisok\*& Bottom, string path, int& countItems);

void delItem(spisok\*& Top, spisok\*& Bottom, int& countItems, int pos);

void replaceInfo(spisok\* Top, int index, int& countItems);

void sortGroup(spisok\* Top, spisok\* Bottom, int& countItems);

void findFio(spisok\* Top, char \_fio[30], int& countItems);

void findGirlsAndBoys(spisok\* Top, int& countItems);

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

spisok\* Top = NULL, \* Bottom = NULL;

int countItems = 0;

int posMenu = 1;

system("cls");

readBinFile(Top, Bottom, "std.dat", countItems);

system("pause");

while (true) {

int s, s1;

system("cls");

printMenu(posMenu);

s = \_getch();

if (s == 224) {

int s1 = \_getch();

if (s1 == 72) {

if (posMenu == 1) posMenu = 10;

else --posMenu;

}

else if (s1 == 80) {

if (posMenu == 10) posMenu = 1;

else ++posMenu;

}

}

else if (s == 13) {

switch (posMenu)

{

case 1: {

system("cls");

string path;

cout << "Введите путь: ";

cin >> path;

readTxtFile(Top, Bottom, path, countItems);

}break;

case 2: {

system("cls");

addItem(Top, Bottom, fillInfo(), countItems);

}break;

case 3: {

system("cls");

printList(Top, countItems);

system("pause");

}break;

case 4: {

system("cls");

findGirlsAndBoys(Top, countItems);

system("pause");

}break;

case 5: {

char fio[30];

system("cls");

cout << "Введите ФИО для поиска: ";

cin >> fio;

while (cin.peek() != '\n') {

char tmp[15];

cin >> tmp;

strcat(fio, " ");

strcat(fio, tmp);

}

findFio(Top, fio, countItems);

system("pause");

}break;

case 6: {

system("cls");

sortGroup(Top, Bottom, countItems);

system("pause");

}break;

case 7: {

system("cls");

int index;

cout << "Введите номер записи для замены: ";

cin >> index;

replaceInfo(Top, index, countItems);

system("pause");

}break;

case 8: {

system("cls");

int pos;

cout << "Введите номер записи: ";

cin >> pos;

delItem(Top, Bottom, countItems, pos);

}break;

case 9: {

system("cls");

string path;

cout << "Введите путь: ";

cin >> path;

writeTxtFile(Top, path, countItems);

}break;

case 10: {

writeBinFile(Top, "std.dat", countItems);

return 0;

}break;

}

}

}

return 0;

}

struct info fillInfo() {

info temp;

cout << "Введите ФИО спортсмена: ";

cin >> temp.fio;

while (cin.peek() != '\n') {

char tmp[15];

cin >> tmp;

strcat(temp.fio, " ");

strcat(temp.fio, tmp);

}

cout << "Введите дату рождения: ";

cin >> temp.birthday.day >> temp.birthday.month >> temp.birthday.year;

cout << "Введите пол [М | Ж]: ";

cin >> temp.gender;

cout << "Введите вес: ";

cin >> temp.weight;

cout << "Введите спортивный разряд: ";

cin >> temp.rank;

while (cin.peek() != '\n') {

char tmp[15];

cin >> tmp;

strcat(temp.rank, " ");

strcat(temp.rank, tmp);

}

cout << "Введите название команды: ";

cin >> temp.team;

while (cin.peek() != '\n') {

char tmp[15];

cin >> tmp;

strcat(temp.team, " ");

strcat(temp.team, tmp);

}

return temp;

}

void addItem(spisok\*& Top, spisok\*& Bottom, info \_info, int& countItems) {

if (Top == NULL) {

spisok\* temp = new spisok;

temp->inf = \_info;

temp->prev = temp;

temp->next = temp;

Top = Bottom = temp;

++countItems;

}

else {

spisok\* temp = new spisok;

temp->inf = \_info;

temp->next = Top;

temp->prev = Bottom;

Bottom->next = temp;

Bottom = temp;

Top->prev = Bottom;

++countItems;

}

}

void printList(spisok\* Top, int& countItems) {

cout << "| ФИО |Д-а рожд. | Пол | Вес | Спортивный разряд | Название команды |" << endl;

cout << "--------------------------------------------------------------------------------------------------------------------" << endl;

spisok\* temp = Top;

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

cout.setf(ios::left);

cout << '"'; cout << setw(29) << temp->inf.fio;

cout << '"'; cout << setw(2) << temp->inf.birthday.day; cout << '.'; cout << setw(2) << temp->inf.birthday.month; cout << '.';

cout << setw(2) << temp->inf.birthday.year;

cout << '"'; cout << setw(5) << temp->inf.gender;

cout << '"'; cout << setw(7) << temp->inf.weight;

cout << '"'; cout << setw(29) << temp->inf.rank;

cout << '"'; cout << setw(29) << temp->inf.team; cout << '"' << endl;

temp = temp->next;

}

}

void writeTxtFile(spisok\* Top, string path, int& countItems) {

if (Top == NULL) {

system("cls");

cout << "Пустой список" << endl;

system("pause");

return;

}

else {

ofstream out(path, ios::out);

spisok\* temp = Top;

for (int i = 0; i < countItems-1; ++i) {

out.setf(ios::left);

out << '"'; out << setw(29) << temp->inf.fio;

out << '"'; out << setw(2) << temp->inf.birthday.day; out << '.'; out << setw(2) << temp->inf.birthday.month; out << '.';

out << setw(2) << temp->inf.birthday.year;

out << '"'; out << setw(5) << temp->inf.gender;

out << '"'; out << setw(7) << temp->inf.weight;

out << '"'; out << setw(29) << temp->inf.rank;

out << '"'; out << setw(29) << temp->inf.team; out << '"' << endl;

temp = temp->next;

}

temp = temp->next;

out.setf(ios::left);

out << '"'; out << setw(29) << temp->inf.fio;

out << '"'; out << setw(2) << temp->inf.birthday.day; out << '.'; out << setw(2) << temp->inf.birthday.month; out << '.';

out << setw(2) << temp->inf.birthday.year;

out << '"'; out << setw(5) << temp->inf.gender;

out << '"'; out << setw(7) << temp->inf.weight;

out << '"'; out << setw(29) << temp->inf.rank;

out << '"'; out << setw(29) << temp->inf.team; out << '"';

out.close();

}

}

void readTxtFile(spisok\*& Top, spisok\*& Bottom, string path, int& countItems) {

ifstream input(path, ios::in);

while (!input.eof()) {

info temp;

char str[118];

input.getline(str, 118);

int i = 1;

char el = str[i++];

while (el != '"') {

temp.fio[i - 2] = el;

el = str[i++];

if ((temp.fio[i - 3] == ' ') && (temp.fio[i - 4] == ' ')) break;

}

temp.fio[i - 4] = '\0';

while (el == ' ' || el == '"') el = str[i++];

int numb = 0;

numb = 0;

while ((el != ' ') && (el != '.')) {

numb = numb \* 10 + (el - '0');

el = str[i++];

}

temp.birthday.day = numb;

el = str[i++];

numb = 0;

while ((el != ' ') && (el != '.')) {

numb = numb \* 10 + (el - '0');

el = str[i++];

}

temp.birthday.month = numb;

el = str[i++];

numb = 0;

while (el == ' ' || el == '.') el = str[i++];

while ((el != ' ') && (el != '"')) {

numb = numb \* 10 + (el - '0');

el = str[i++];

}

temp.birthday.year = numb;

el = str[i++];

temp.gender = el;

el = str[i++];

while (el == ' ' || el == '"') el = str[i++];

char temp1[7];

while ((el != ' ') && (el != '"')) {

temp1[i - 49] = el;

el = str[i++];

}

temp1[i - 49] = '\0';

temp.weight = atof(temp1);

while (el == ' ' || el == '"') el = str[i++];

while (el != '"') {

temp.rank[i - 57] = el;

el = str[i++];

}

temp.rank[i - 57] = '\0';

while (el == ' ' || el == '"') el = str[i++];

while (el != '"') {

temp.team[i - 87] = el;

el = str[i++];

if ((temp.team[i - 88] == ' ') && (temp.team[i - 89] == ' ')) break;

}

temp.team[i - 89] = '\0';

addItem(Top, Bottom, temp, countItems);

}

input.close();

}

void printMenu(int& pos) {

if (pos == 1) {

cout << "==> " << "Создание из текстового файла" << endl;

cout << " " << "Добавление элемента в список" << endl;

cout << " " << "Вывод списка на экран" << endl;

cout << " " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << " " << "Поиск по ФИО" << endl;

cout << " " << "Сортировка списка по командам " << endl;

cout << " " << "Редактирование записи" << endl;

cout << " " << "Удаление записи" << endl;

cout << " " << "Запись списка в текстовый файл" << endl;

cout << " " << "Выход" << endl;

}

else if (pos == 2) {

cout << " " << "Создание из текстового файла" << endl;

cout << "==> " << "Добавление элемента в список" << endl;

cout << " " << "Вывод списка на экран" << endl;

cout << " " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << " " << "Поиск по ФИО" << endl;

cout << " " << "Сортировка списка по командам " << endl;

cout << " " << "Редактирование записи" << endl;

cout << " " << "Удаление записи" << endl;

cout << " " << "Запись списка в текстовый файл" << endl;

cout << " " << "Выход" << endl;

}

else if (pos == 3) {

cout << " " << "Создание из текстового файла" << endl;

cout << " " << "Добавление элемента в список" << endl;

cout << "==> " << "Вывод списка на экран" << endl;

cout << " " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << " " << "Поиск по ФИО" << endl;

cout << " " << "Сортировка списка по командам " << endl;

cout << " " << "Редактирование записи" << endl;

cout << " " << "Удаление записи" << endl;

cout << " " << "Запись списка в текстовый файл" << endl;

cout << " " << "Выход" << endl;

}

else if (pos == 4) {

cout << " " << "Создание из текстового файла" << endl;

cout << " " << "Добавление элемента в список" << endl;

cout << " " << "Вывод списка на экран" << endl;

cout << "==> " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << " " << "Поиск по ФИО" << endl;

cout << " " << "Сортировка списка по командам " << endl;

cout << " " << "Редактирование записи" << endl;

cout << " " << "Удаление записи" << endl;

cout << " " << "Запись списка в текстовый файл" << endl;

cout << " " << "Выход" << endl;

}

else if (pos == 5) {

cout << " " << "Создание из текстового файла" << endl;

cout << " " << "Добавление элемента в список" << endl;

cout << " " << "Вывод списка на экран" << endl;

cout << " " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << "==> " << "Поиск по ФИО" << endl;

cout << " " << "Сортировка списка по командам " << endl;

cout << " " << "Редактирование записи" << endl;

cout << " " << "Удаление записи" << endl;

cout << " " << "Запись списка в текстовый файл" << endl;

cout << " " << "Выход" << endl;

}

else if (pos == 6) {

cout << " " << "Создание из текстового файла" << endl;

cout << " " << "Добавление элемента в список" << endl;

cout << " " << "Вывод списка на экран" << endl;

cout << " " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << " " << "Поиск по ФИО" << endl;

cout << "==> " << "Сортировка списка по командам " << endl;

cout << " " << "Редактирование записи" << endl;

cout << " " << "Удаление записи" << endl;

cout << " " << "Запись списка в текстовый файл" << endl;

cout << " " << "Выход" << endl;

}

else if (pos == 7) {

cout << " " << "Создание из текстового файла" << endl;

cout << " " << "Добавление элемента в список" << endl;

cout << " " << "Вывод списка на экран" << endl;

cout << " " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << " " << "Поиск по ФИО" << endl;

cout << " " << "Сортировка списка по командам " << endl;

cout << "==> " << "Редактирование записи" << endl;

cout << " " << "Удаление записи" << endl;

cout << " " << "Запись списка в текстовый файл" << endl;

cout << " " << "Выход" << endl;

}

else if (pos == 8) {

cout << " " << "Создание из текстового файла" << endl;

cout << " " << "Добавление элемента в список" << endl;

cout << " " << "Вывод списка на экран" << endl;

cout << " " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << " " << "Поиск по ФИО" << endl;

cout << " " << "Сортировка списка по командам " << endl;

cout << " " << "Редактирование записи" << endl;

cout << "==> " << "Удаление записи" << endl;

cout << " " << "Запись списка в текстовый файл" << endl;

cout << " " << "Выход" << endl;

}

else if (pos == 9) {

cout << " " << "Создание из текстового файла" << endl;

cout << " " << "Добавление элемента в список" << endl;

cout << " " << "Вывод списка на экран" << endl;

cout << " " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << " " << "Поиск по ФИО" << endl;

cout << " " << "Сортировка списка по командам " << endl;

cout << " " << "Редактирование записи" << endl;

cout << " " << "Удаление записи" << endl;

cout << "==> " << "Запись списка в текстовый файл" << endl;

cout << " " << "Выход" << endl;

}

else if (pos == 10) {

cout << " " << "Создание из текстового файла" << endl;

cout << " " << "Добавление элемента в список" << endl;

cout << " " << "Вывод списка на экран" << endl;

cout << " " << "Нахождение для каждой группы 5 старших девушек и 5 тяжелых юношей" << endl;

cout << " " << "Поиск по ФИО" << endl;

cout << " " << "Сортировка списка по командам " << endl;

cout << " " << "Редактирование записи" << endl;

cout << " " << "Удаление записи" << endl;

cout << " " << "Запись списка в текстовый файл" << endl;

cout << "==> " << "Выход" << endl;

}

}

void writeBinFile(spisok\* Top, string path, int& countItems) {

if (Top == NULL) {

system("cls");

cout << "Список пуст" << endl;

system("pause");

return;

}

else {

ofstream out(path, ios::binary);

spisok\* temp = Top;

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

out.write((const char\*) & (temp->inf), sizeof(info));

temp = temp->next;

}

out.close();

}

}

void readBinFile(spisok\*& Top, spisok\*& Bottom, string path, int& countItems) {

ifstream input(path, ios::binary);

if (!input) {

cout << "Возникла ошибка при открытии бинарного фала" << endl;

return;

}

while (!input.eof()) {

info temp;

input.read((char\*)&temp, sizeof(info));

addItem(Top, Bottom, temp, countItems);

}

input.close();

delItem(Top, Bottom, countItems, countItems);

}

void delItem(spisok\*& Top, spisok\*& Bottom, int& countItems, int pos) {

if (Top == NULL) {

system("cls");

cout << "Пустой список" << endl;

system("pause");

return;

}

else if (pos > countItems) {

cout << "Нет такой записи" << endl;

system("pause");

return;

}

else {

spisok\* temp = Top;

for (int i = 1; i < pos; ++i) temp = temp->next;

if (countItems == 1) {

delete temp;

Top = Bottom = NULL;

--countItems;

}

else if (countItems == 2) {

temp->prev->next = temp->prev;

temp->prev->prev = temp->prev;

Top = Bottom = temp->prev;

delete temp;

--countItems;

}

else {

if (temp == Top) {

Top->next->prev = Bottom;

Bottom->next = Top->next;

delete temp;

Top = Bottom->next;

--countItems;

}

else if (temp == Bottom) {

Bottom->prev->next = Top;

Top->prev = Bottom->prev;

delete Bottom;

Bottom = Top->prev;

--countItems;

}

else {

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

delete temp;

--countItems;

}

}

}

}

void replaceInfo(spisok\* Top, int index, int& countItems) {

if (Top == NULL) {

cout << "Пустой список" << endl;

return;

}

else if (index > countItems || index < 1) {

cout << "Нет такой записи" << endl;

return;

}

else {

cout << "Замена содержимого записи" << endl;

spisok\* temp = Top;

for (int i = 1; i < index; ++i) temp = temp->next;

info tempInfo;

tempInfo = fillInfo();

temp->inf = tempInfo;

}

}

void sortGroup(spisok\* Top, spisok\* Bottom, int& countItems) {

if (Top == NULL) {

cout << "Пустой список" << endl;

system("pause");

return;

}

else {

spisok\* temp = Top;

for (int i = 0; i < countItems - 2; ++i) {

spisok\* temp1 = temp->next;

for (int j = i + 1; j < countItems; ++j) {

if ((strcmp(temp->inf.team, temp1->inf.team) > 0)) {

info tmp = temp->inf;

temp->inf = temp1->inf;

temp1->inf = tmp;

}

temp1 = temp1->next;

}

temp = temp->next;

}

}

cout << "Сортировка выполнена" << endl;

}

void findFio(spisok\* Top, char \_fio[30], int& countItems) {

if (Top == NULL) {

cout << "Пустой список" << endl;

system("pause");

return;

}

spisok\* temp = Top;

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

if (strcmp(temp->inf.fio, \_fio) == 0) {

cout.setf(ios::left);

cout << '"'; cout << setw(29) << temp->inf.fio;

cout << '"'; cout << setw(2) << temp->inf.birthday.day; cout << '.'; cout << setw(2) << temp->inf.birthday.month; cout << '.';

cout << setw(2) << temp->inf.birthday.year;

cout << '"'; cout << setw(5) << temp->inf.gender;

cout << '"'; cout << setw(7) << temp->inf.weight;

cout << '"'; cout << setw(29) << temp->inf.rank;

cout << '"'; cout << setw(29) << temp->inf.team; cout << '"' << endl;

}

temp = temp->next;

}

}

void findGirlsAndBoys(spisok\* Top, int& countItems) {

if (Top == NULL) {

cout << "Пустой список" << endl;

return;

}

int count = 1;

spisok\* temp = Top->next;

char teams[100][30];

strcpy(teams[0], Top->inf.team);

for (int i = 0; i < countItems - 1; ++i) {

bool find = false;

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

if (strcmp(temp->inf.team, teams[j]) == 0) {

find = true;

}

if (find) {

break;

}

else continue;

}

if (!find) {

strcpy(teams[count++], temp->inf.team);

}

temp = temp->next;

}

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

int countGirls = 0;

info tempGroupGirls[100];

temp = Top;

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

if ((strcmp(temp->inf.team, teams[i]) == 0) && (temp->inf.gender == 'Ж')) {

tempGroupGirls[countGirls++] = temp->inf;

}

temp = temp->next;

}

for (int j = 0; j < countGirls - 1; ++j) {

for (int h = j + 1; h < countGirls; ++h) {

if (tempGroupGirls[h].birthday.year < tempGroupGirls[j].birthday.year || ((tempGroupGirls[h].birthday.year == tempGroupGirls[j].birthday.year)

&& (tempGroupGirls[h].birthday.month < tempGroupGirls[j].birthday.month)) || ((tempGroupGirls[h].birthday.year == tempGroupGirls[j].birthday.year)

&& (tempGroupGirls[h].birthday.month == tempGroupGirls[j].birthday.month) && (tempGroupGirls[h].birthday.day < tempGroupGirls[j].birthday.day))) {

info tmp = tempGroupGirls[j];

tempGroupGirls[j] = tempGroupGirls[h];

tempGroupGirls[h] = tmp;

}

}

}

int countMans = 0;

info tempGroupMans[100];

temp = Top;

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

if ((strcmp(temp->inf.team, teams[i]) == 0) && (temp->inf.gender == 'М')) {

tempGroupMans[countMans++] = temp->inf;

}

temp = temp->next;

}

for (int j = 0; j < countMans - 1; ++j) {

for (int h = j + 1; h < countMans; ++h) {

if (tempGroupMans[h].weight > tempGroupMans[j].weight) {

info tmp = tempGroupMans[j];

tempGroupMans[j] = tempGroupMans[h];

tempGroupMans[h] = tmp;

}

}

}

cout << "Группа " << teams[i] << endl;

if (countGirls == 0) {

cout << " Нет девушек в команде" << endl;

}

else {

cout << " Девушки:" << endl;

for (int k = 0; k < countGirls; ++k) {

if (k > 4) break;

cout.setf(ios::left);

cout << " ";

cout << '"'; cout << setw(29) << tempGroupGirls[k].fio;

cout << '"'; cout << setw(2) << tempGroupGirls[k].birthday.day; cout << '.'; cout << setw(2) << tempGroupGirls[k].birthday.month; cout << '.';

cout << setw(2) << tempGroupGirls[k].birthday.year;

cout << '"'; cout << setw(5) << tempGroupGirls[k].gender;

cout << '"'; cout << setw(7) << tempGroupGirls[k].weight;

cout << '"'; cout << setw(29) << tempGroupGirls[k].rank;

cout << '"'; cout << setw(29) << tempGroupGirls[k].team; cout << '"' << endl;

}

}

if (countMans == 0) {

cout << " Нет юношей в команде" << endl;

}

else {

cout << " Юноши:" << endl;

for (int k = 0; k < countMans; ++k) {

if (k > 4) break;

cout.setf(ios::left);

cout << " ";

cout << '"'; cout << setw(29) << tempGroupMans[k].fio;

cout << '"'; cout << setw(2) << tempGroupMans[k].birthday.day; cout << '.'; cout << setw(2) << tempGroupMans[k].birthday.month; cout << '.';

cout << setw(2) << tempGroupMans[k].birthday.year;

cout << '"'; cout << setw(5) << tempGroupMans[k].gender;

cout << '"'; cout << setw(7) << tempGroupMans[k].weight;

cout << '"'; cout << setw(29) << tempGroupMans[k].rank;

cout << '"'; cout << setw(29) << tempGroupMans[k].team; cout << '"' << endl;

}

}

cout << endl;

}

}