**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

**Курсовая работа**

**по дисциплине «Программирование»**

**Тема: Разработка электронной картотеки**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3312 |  | Шарапов И. Д. |
| Преподаватель |  | Аббас С. А. |

Санкт-Петербург

2024

**Содержание**

[Цель работы 3](#_Toc167790855)

[Задание 3](#_Toc167790856)

[Постановка задачи и описание решения 4](#_Toc167790857)

[Описание структур 6](#_Toc167790858)

[Описание функций 6](#_Toc167790859)

[Структура вызова функций 8](#_Toc167790860)

[Описание переменных 8](#_Toc167790861)

[Схема алгоритма 12](#_Toc167790862)

[Текст программы 12](#_Toc167790863)

[Контрольные примеры 22](#_Toc167790864)

[Содержимое файлов 23](#_Toc167790865)

[Примеры выполнения программы 23](#_Toc167790866)

[Выводы 23](#_Toc167790867)

# Цель работы

Полное решение содержательной задачи (содержательная и формальная постановка задачи, спецификация, включая описание диалога, выбор метода решения и структур данных, разработка алгоритма, программная реализация, тестирование и отладка, документирование). Создание электронной картотеки спортсменов.

# Задание

Создать электронную картотеку, хранящуюся на диске, и программу на языке Си, обеспечивающую взаимодействие с ней. Программа выполняет:

* занесение данных в электронную картотеку;
* внесение изменений (исключение, корректировка, добавление);
* поиск данных по различным признакам;
* сортировку по различным признакам;
* вывод результатов на экран и сохранение на диске.

Выбор подлежащих выполнению команд должен быть реализован с помощью основного меню и вложенных меню.

Задача должна быть структурирована и отдельные части должны быть оформлены как функции.

Исходные данные должны вводиться с клавиатуры. В процессе обработки картотека должна храниться в памяти компьютера в виде списков и массивов структур, связанных указателями. Типы списков и структур выбираются исходя из предметной области.

Картотека составляется по выбранной предметной области.

В программе должно быть реализовано простейшее меню. Выполнение программы должно быть многократным по желанию пользователя. Данные первоначально считываются из файла (файлов), в процессе работы данные вводятся с клавиатуры.

Перечень пунктов меню:

1. Вывод картотеки. (В консоль в виде таблички);
2. Поиск карточек по параметру (Выбор параметра и последующий поиск по строке);
3. Сортировка картотеки по параметру;
4. Добавление карточки спортсмена (Ввод строки в формате);
5. Редактирование карточки по ID (Последовательный ввод полей, либо пропуск);
6. Удаление карточек по параметру (Поиск подходящих строк и подтверждение их удаления);
7. Сохранение на диск;
8. Выход из программы.

# Постановка задачи и описание решения

Для решения задачи необходимо написать программу, которая взаимодействует с электронной картотекой. Для этого используется структура Athlete, в которой содержится 7 полей: имя спортсмена; университет, в котором учится спортсмен; возраст; вес; рост; массив из трёх чисел (результаты выступления спортсмена); индекс результатов спортсмена (отношение суммы к весу).

Вначале программа спрашивает у пользователя из какого файла взять первичную информацию. Далее выводится главное меню (список всех команд, доступных пользователю), для этого программа обращается к функции *help()*. Список команд, которые она выводит:

*!print* – вывод всех карточек в виде таблице. Программа обращается к функции *print()*.

*!find* – поиск карточек по параметру. Программа вызывает функцию *find()*. Пользователю предлагается выбрать один из 10 параметров для сортировки, либо 0 для выхода в главное меню. Если пользователь выбрал не 0, то у него спрашивается подстрока, которая будет искаться без учёта регистра. Пользователю выводится все найденные пользователи, либо сообщение *«No matches found!»*. В первом случае пользователю предлагается сортировать найденные данные по всем параметрам, при этом исходный список не будет меняться. Для этого используется функция *sorted()*.

*!sort* - сортировка карточек по параметру. Программа вызывает функцию *sort()*. Пользователю предлагается сортировать данные по любому параметру, до тех пор, пока он не введёт 0. При этом данные в памяти компьютера тоже меняют своё положение.

*!add* – добавление карточки. Программа вызывает функцию *add()*. Пользователю выводится формат, в котором он должен ввести данные о новом спортсмене, далее спрашивается сама строка. Программа обрабатывает данную строку и добавляет спортсмена в конец списка.

*!edit* – изменение конкретной карточки. Программа вызывает функцию *edit()*. Далее у пользователя спрашивается ID спортсмена, которого он хочет изменить. Если спортсмен с данным ID существует, пользователю предлагается последовательно менять поля этого спортсмена. Если пользователь не хочет менять конкретное поле, он может нажать *ENTER*.

*!delete* – удаление карточек по параметру. Программа вызывает функцию *delete()*. Как и в *find()* выбираются спортсмены и выводятся пользователю. Далее у пользователя спрашивается подтверждение: точно ли он хочет удалить их. Если пользователь ответит *Y*, то данные об этих пользователях удаляются, а память очищается. В конце пользователю выводится изменённый список.

*!save* – сохранение данных в файл. Программа вызывает функцию *save()*. У пользователя спрашивается имя файла, в который он хочет сохранить данные. После считывания строки данные сохраняются в этот файл в формате CSV.

*!end* – завершает выполнение программы и очищает память.

При переходе из главного меню к конкретному функционалу экран очищается, и пользователю для наглядности выводится вся табличка. В конце выполнения функции экран также очищается, и пользователь «попадает» в главное меню, где уже сразу отображён список всех возможных команд.

# Описание структур

1. Структура Athlete

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя переменной** | **Тип** | **Назначение** |
| 1 | name | char\* | Имя спортсмена |
| 2 | university | char\* | Название университета |
| 3 | age | int | Возраст |
| 4 | weight | float | Вес в килограммах |
| 5 | height | int | Рост в сантиметрах |
| 6 | result | int[3] | Результаты выступления |
| 7 | index | float | Отношение результата к весу |

2. Структура NodeOfList

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя переменной** | **Тип** | **Назначение** |
| 1 | id | int | Уникальный ID |
| 2 | data | Athlete\* | Указатель на данные о спортсмене |
| 3 | next | struct NodeOfList\* | Указатель на следующую вершину списка |
| 4 | prev | struct NodeOfList\* | Указатель на предыдущую вершину списка |

3. Структура ListOfAthlete

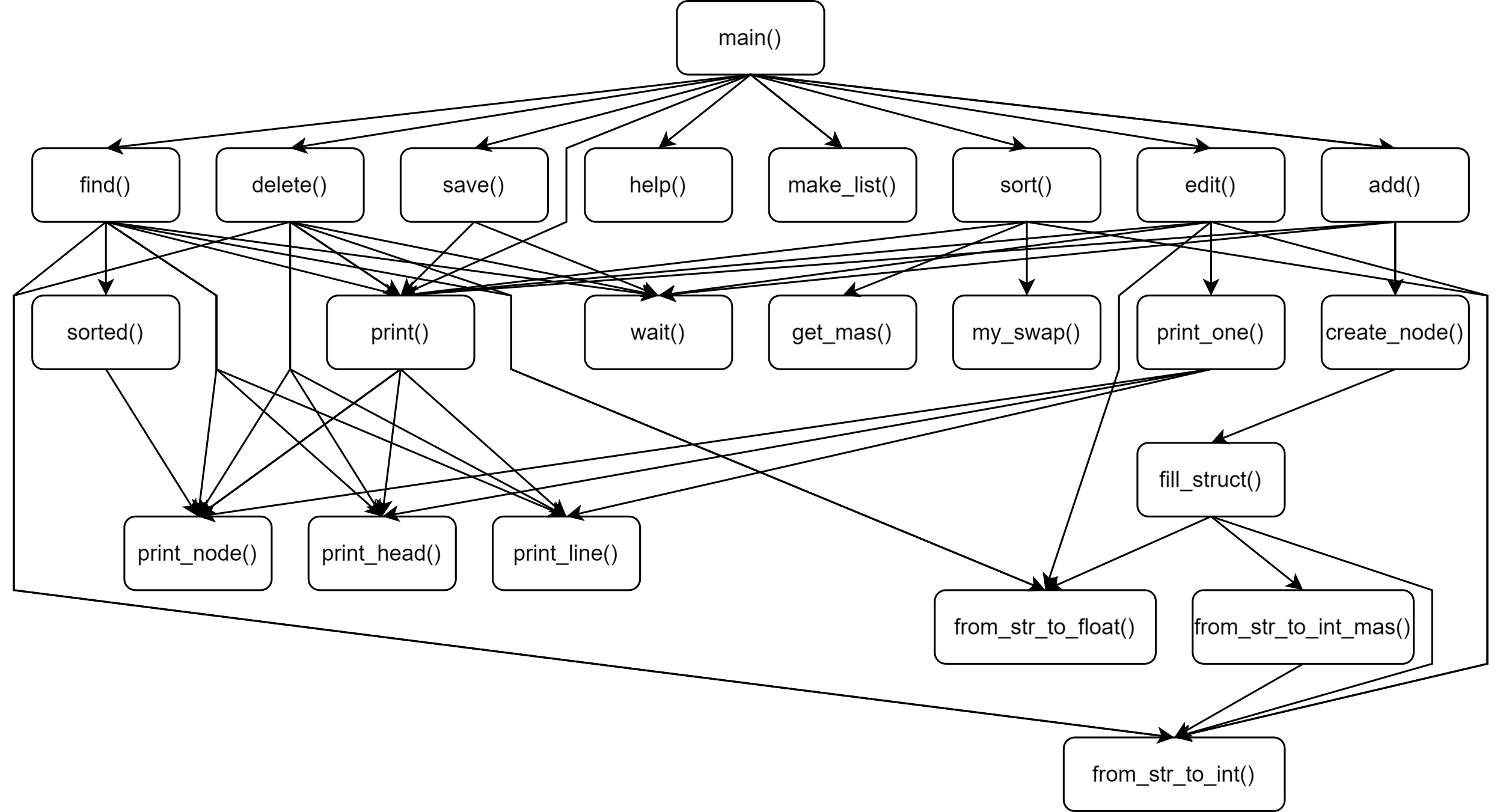
|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя переменной** | **Тип** | **Назначение** |
| 1 | length | int | Уникальный ID |
| 2 | first | struct NodeOfList\* | Указатель на первую вершину списка |
| 3 | last | struct NodeOfList\* | Указатель на последнюю вершину списка |

# Описание функций

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Назначение** |
| 1 | main | Основная функция программы. Открывает файл, инициализирует список, отвечает за взаимодействие с пользователем через меню. В конце очищает память. |
| 2 | from\_str\_to\_int | Конвертирует строку в целочисленное значение. Если символ не цифра – возвращает 0. |
| 3 | from\_str\_to\_float | Конвертирует строку в число с плавающей точкой. Если символ не цифра – возвращает 0. |
| 4 | from\_str\_to\_int\_mas | Конвертирует строку в массив целочисленных значений длины 3. |
| 5 | fill\_struct | Извлекает данные из строки и заполняет поля структуры Athlete. |
| 6 | make\_list | Создаёт список (выделяет память и возвращает указатель на него). |
| 7 | create\_node | Выделяет память под новую вершину списка, заполняет его данными и возвращает указатель на него. |
| 8 | help | Отображает доступные команды пользователю в консоли. |
| 9 | wait | Приостанавливает выполнение программы до нажатия клавиши Enter. |
| 10 | print\_line | Выводит линию для разделения таблицы данных в консоли. |
| 11 | print\_head | Выводит поля заголовка таблицы в консоль. |
| 12 | print\_node | Выводит данные одного элемента списка в консоль. |
| 13 | print\_one | Выводит данные одного элемента списка с заголовком в консоль. |
| 14 | print | Выводит все элементы списка в консоль. |
| 15 | sorted | Сортировка выбранных элементов списка и вывод отсортированных данных в консоль. (не влияет на порядок основного списка) |
| 16 | find | Поиск элементов списка и вывод найденных данных в консоль с возможностью последующей сортировки этих данных. |
| 17 | get\_mas | Функция проходит по всему списку и заполняет массив указателями на вершины списка. Возвращает массив указателей. |
| 18 | my\_swap | Меняет два значения в списке местами. |
| 19 | sort | Сортирует элементы списка в соответствии с заданным параметром и выводит отсортированные данные в консоль. |
| 20 | add | Пользователь вводит данные нового элемента, функция создаёт новою вершину и добавляет её в конец списка. |
| 21 | edit | Пользователь вводит ID элемента, который хочет отредактировать, и новые данные. Функция находит элемент по ID и предлагает изменить его данные по отдельным полям. |
| 22 | delete | Пользователь выбирает поле для удаления и вводит критерии поиска. Функция удаляет элементы, соответствующие заданным критериям. |

|  |  |  |
| --- | --- | --- |
| 23 | save | Пользователь выбирает имя файла, в который будут сохранены данные списка. Функция записывает все данные в файл в соответствии с форматом. |

# Структура вызова функций



# Описание переменных

|  |  |  |  |
| --- | --- | --- | --- |
| Функция *int main()* | | | |
| № | Имя переменной | Тип | Назначение |
| 1 | list | ListOfAthlete\* | Список спортсменов |
| 2 | g\_id | int | Глобальный ID |
| 3 | cl | int | Флаг для отображения help |
| 4 | filename | char[] | Буфер для хранения имени файла |
| 5 | str | char[] | Буфер для команд пользователя |
| 6 | text | char[] | Буфер для содержимого файла |
| 7 | cur\_node | NodeOfList\* | Текущий элемент в списке |
| 8 | f | FILE\* | Указатель на файл |
| Функция *int from\_str\_to\_int(char \*str)* | | | |
| 1 | str | char\* | Стока, которую нужно конвертировать в число |
| 2 | ans | int | Результирующее число |
| 3 | x | int | Текущая цифра |

|  |  |  |  |
| --- | --- | --- | --- |
| Функция *float from\_str\_to\_float(char \*str)* | | | |
| 1 | str | char\* | Стока, которую нужно конвертировать в число с плавающей точкой |
| 2 | ans | float | Результирующее число с плавающей точкой |
| 3 | a | float | Целая часть числа |
| 4 | b | float | Дробная часть числа |
| 5 | x | float | Текущая цифра |
| Функция *void from\_str\_to\_int\_mas(char \*str, int \*mas)* | | | |
| 1 | str | char\* | Стока, которую нужно конвертировать в массив чисел длины 3 |
| 2 | mas | int\* | Указатель на массив, в который будут записаны числа |
| 3 | ind | int | Текущий индекс в массиве |
| 4 | j | int | Текущий индекс в строке |
| Функция *Athlete \*fill\_struct(char \*str)* | | | |
| 1 | str | char\* | Строка, которую нужно конвертировать в сущность Athlete |
| 2 | user | Athlete\* | Сущность Athlete |
| 3 | word | char\* | Текущая подстрока в строке |
| 4 | pole | char\*[] | Массив указателей на подстроки |
| 5 | ind | int | Текущий индекс в массиве подстрок |
| 6 | tt | int | Текущий индекс в строке |
| Функция *ListOfAthlete \*make\_list()* | | | |
| 1 | ph | ListOfAthlete\* | Указатель на создаваемый список |
| Функция *NodeOfList \*create\_node(char \*str, int g\_id)* | | | |
| 1 | str | char\* | Строка, которую нужно конвертировать в сущность Athlete |
| 2 | g\_id | int | Глобальный ID |
| 3 | new\_node | NodeOfList\* | Указатель на создаваемую вершину |
| Функция *void print\_node(NodeOfList \*node)* | | | |
| 1 | node | NodeOfList\* | Вершина списка, которую нужно вывести |
| Функция *void print\_one(NodeOfList \*node)* | | | |
| 1 | node | NodeOfList\* | Вершина списка, которую нужно вывести |
| Функция *void print(ListOfAthlete \*list)* | | | |
| 1 | list | ListOfAthlete\* | Список, который нужно вывести |
| Функция *void sorted(int \*mas, ListOfAthlete \*list, int param)* | | | |
| 1 | mas | int\* | Массив флагов для сортировки |
| 2 | list | ListOfAthlete\* | Список спортсменов |
| 3 | param | int | Параметр, по которому сортируем |
| 4 | cur\_node | NodeOfList\* | Текущая вершина списка |
| 5 | min\_node | NodeOfList\* | Минимальная вершина списка |
| 6 | ind | int | Индекс минимальной вершины |

|  |  |  |  |
| --- | --- | --- | --- |
| Функция *void find(ListOfAthlete \*list)* | | | |
| 1 | list | ListOfAthlete\* | Список спортсменов |
| 2 | cur\_node | NodeOfList\* | Текущая вершина списка |
| 3 | str | char[] | Строка, которую ищет пользователь |
| 4 | mas | int[] | Массив флагов для поиска |
| 5 | fl | int | Флаг на то, что хоть один элемент найден |
| 6 | param | int | Параметр, по которому ищем |
| Функция *NodeOfList \*\*get\_mas(ListOfAthlete \*list)* | | | |
| 1 | list | ListOfAthlete\* | Список спортсменов |
| 2 | cur\_node | NodeOfList\* | Текущая вершина списка |
| 3 | mas | NodeOfList\*\* | Массив указателей на вершины списка |
| Функция *void my\_swap(NodeOfList \*\*mas, ListOfAthlete \*list, int i, int j)* | | | |
| 1 | mas | NodeOfList\*\* | Массив указателей на вершины списка |
| 2 | list | ListOfAthlete\* | Список спортсменов |
| 3 | i | int | Индекс первого узла |
| 4 | j | int | Индекс второго узла |
| 5 | q | NodeOfList\* | Вспомогательная вершина |
| Функция *void sort(ListOfAthlete \*list)* | | | |
| 1 | list | ListOfAthlete\* | Список спортсменов |
| 2 | mas | NodeOfList\*\* | Массив указателей на вершины списка |
| 3 | str | char[] | Параметр в виде строки |
| 4 | n | int | Количество вершин в списке |
| 5 | param | int | Параметр, по которому сортируем |
| Функция *void add(ListOfAthlete \*list, int g\_id)* | | | |
| 1 | list | ListOfAthlete\* | Список спортсменов |
| 2 | g\_id | int | Глобальный ID |
| 3 | str | char[] | Буфер для данных о спортсмене |
| 4 | cur\_node | NodeOfList\* | Текущая вершина списка |
| Функция *void edit(ListOfAthlete \*list)* | | | |
| 1 | list | ListOfAthlete\* | Список спортсменов |
| 2 | the\_node | NodeOfList\* | Вершина, которую редактируем |
| 3 | id | int | ID спортсмена для редактирования |
| 4 | str | char[] | Буфер для пользовательского ввода |
| 5 | s\_id | char[] | Буфер ID спортсмена в строковом виде |

|  |  |  |  |
| --- | --- | --- | --- |
| Функция *void delete(ListOfAthlete \*list)* | | | |
| 1 | list | ListOfAthlete\* | Список спортсменов |
| 2 | cur\_node | NodeOfList\* | Текущая вершина списка |
| 3 | prev\_node | NodeOfList\* | Предыдущая вершина списка |
| 4 | str | char[] | Строка, которую ищет пользователь |
| 5 | ch | char | Символ для подтверждения |
| 6 | mas | int[] | Массив флагов для удаления |
| 7 | fl | int | Флаг на то, что хоть один элемент найден |
| 8 | param | int | Параметр, по которому ищем |
| 9 | cnt | int | Количество элементов для удаления |
| Функция *void save(ListOfAthlete \*list)* | | | |
| 1 | list | ListOfAthlete\* | Список спортсменов |
| 2 | f | FILE\* | Указатель на файл |
| 3 | filename | char[] | Буфер для имени файла |

# **Схема алгоритма**

# Текст программы

|  |
| --- |
|  |

# Контрольные примеры

|  |  |  |
| --- | --- | --- |
| № | Исходные данные | Результаты |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

# Содержимое файлов

# Примеры выполнения программы

# Выводы

Создана электронная картотека и программа на языке Си, обеспечивающая взаимодействие с ней. Реализованы все указанные в задании функции. Создано понятное пользователю меню. Программа работоспособна и протестирована на Windows и Linux.

В работе использованы следующие заголовочные файлы стандартной библиотеки:

* *<stdio.h>* - используется для ввода и вывода из файла и консоли.
* *<stdlib.h>* - используется для выделения памяти для списка и динамических массивов.
* *<string.h>* - используется для обработки и преобразования строк.
* *<math.h>* - используется для поиска модуля числа.