МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

(факультет)

Кафедра Систем управления и информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине: «Основы программирования и алгоритмизации» .

Тема: «Реализация программы оценки соревнований по велоспорту» .

Расчетно-пояснительная записка

Разработал студент Е.Р.Падалка

Подпись, дата Инициалы, фамилия

Руководитель О.В. Минакова

Подпись, дата Инициалы, фамилия

Нормоконтролер

Подпись, дата Инициалы, фамилия

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

ВОРОНЕЖ

2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Кафедра графики, конструирования и информационных технологий в промышленном дизайне

ЗАДАНИЕ

на курсовой проект

по дисциплине: «Основы программирования и алгоритмизации» .

Тема:« Реализация программы оценки соревнований по велоспорту»

Студент бИСТ-224 Падалка Евгений Романович .

Группа, фамилия, имя, отчество

Вариант: Массив структур N спортсменов с фиксированными столбцами, сортировка по среднему значению из 5 этапов, поиск по штрафным очкам в выбираемом диапазоне.

Технические условия:. процессор: AMD(R) Ryzen 5(TM) 4600h CPU @ 3GHz, AMD Radeon graphics, ОС:Windows 11, ОЗУ: 8.00 ГБ 64-разр .

Содержание и объем работы (графические работы, расчеты и прочее):. .

. 36 стр., 33 иллюстр., 4 табл. .

Сроки выполнения этапов:.анализ и постановка задачи (01.10-15.10); разработка пошаговой детализации программы (16.10-10.11); реализация программы (11.11-01.12); тестирование программы (01.12-10.12); оформление пояснительной записки (02.12-12.12).

Срок защиты курсового проекта: .

Руководитель . О.В. Минакова .

Подпись, дата Инициалы, фамилия

Задание принял студент . . Е.Р.Падалка

Подпись, дата Инициалы, фамилия

Замечания руководителя

СОДЕРЖАНИЕ

Введение…………………………………………………………………………...5

1 Постановка задачи……………………………………………………………....6

2 Конструирование программы…………………………………………………11

3Тестирование программы……………………………………………………...22

Заключение……………………………………………………………………….24

Список использованных источников…………………………………………...26

Приложения ……….…….…….……..………………………..………………...27

ВВЕДЕНИЕ

На сегодняшний день очень важно начинающему программисту овладеть навыками работы с функциями, а также работы с массивами, указателями и структурами. Можно сказать, без качественных знаний в этих областях человека, занимающегося программированием, назвать хорошим специалистом будет трудно.

Нередко случается, что какая-то часть программы (блок кода) неоднократно повторяется. Чтобы устранить подобного рода избыточность программного кода, используют понятие функции. Функции - фрагменты программного кода, которые по вызову возвращают значение по каким-либо правилам. Плюсами функций являются также собственная область видимости, возможность процесса рекурсии и передача в другие функции.

Хороший тон – не «засорять» главную функцию – «main», поэтому широко используется деление программы на модули, состоящие из отдельных функций, структур и так далее.

Таким образом, не придётся многократно воспроизводить код в главной функции, создавать большое число переменных, их инициализировать и так далее. Всего один фрагмент, который позволит написать вместо тысячи строк в главной функции всего сто.

**Цель работы:** разработка программы формирования данных о проведении соревнований по велоспорту.

Для достижения поставленной цели, необходимо решить следующие задачи:

* Обосновать выбор структуры данных для обработки таблицы результатов соревнований.
* Реализовать программу с функциями считывания, обработки, вывода, записи информации. Сортировки составленной таблицы по очкам и изменения считанных в нее данных. А также вывода в консоль команд по диапазону очков.
* Протестировать разработанную программу.

1 Постановка задачи

Суть соревнований заключается в прохождении установленной дистанции всеми участниками. Гонщики стартуют одновременно, стараясь преодолеть дистанцию как можно быстрее. О начале соревнований участников извещает специальный сигнал судьи. После него велосипедисты могут стартовать и двигаться по дистанции к финишной отметке.

Гонка ведется спортсменами по улицам города по кругу (кольцевая гонка). Длина одного круга – от одного до трёх километров, количество таких кругов — до пятидесяти (в нашем случае – пять). Длина круга и количество кругов в гонке может сильно варьироваться в зависимости от решения оргкомитета гонки и судейства. По окончанию каждого этапа(круга) участники занимают определенные места, за что получают соответствующие очки. Участник, который получает наибольшее количество очков, побеждает в общем зачете соревнования.

Во время прохождения соревнования игроки могут получать штрафные очки за различные нарушения, которые не разделяются на этапы, и имеют значения лишь на генеральном этапе.

В случае одинаковых баллов N игроков, они занимают одинаковое место M, а следующий за ними по баллам игрок, занимает место M+N, где N - количество игроков, а M - занимаемое место

Таблица 1. Исходные данные

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Исходная таблица** | | | | | | | |
| Номер | Фамилия Имя | Этап1 | Этап2 | Этап3 | Этап4 | Этап5 | Штрафные очки |
| 1 | Geoffrey BOUCHARD | 5 | 0 | 2 | 0 | 0 | 3 |
| 2 | Phil BAUHAUS | 0 | 0 | 0 | 5 | 0 | 1 |
| 3 | Leonardo BASSO | 1 | 0 | 0 | 2 | 1 | 2 |

Таблица 2. Пример вычисляемых значений

|  |  |  |
| --- | --- | --- |
| **Результирующая таблица** | | |
| Фамилия Имя | Общие очки | Место |
| Geoffrey BOUCHARD | 4 | 4 |
| Phil BAUHAUS | 4 | 4 |
| Leonardo BASSO | 2 | 6 |

Задачей курсового проекта является реализация программы «табличный процессор» по теме «соревнования по велоспорту». Тематика задачи подразумевает следующие поля ввода:

* + Номер участника – числовой код, содержащий исключительно цифры
  + Имя и фамилия – строка, содержащая пробелы и возможно, другие разделители (!, - , )
  + Этапы – число, которое принимает значения от 0 до условленного верхнего предела
  + Среднее значение этапов – так как нам необходимо произвести деление, необходимо использовать значение с плавающей запятой, достаточно одного знака после запятой.
  + Штрафные очки – целое число, принимающее значения от 0 до 10
  + Общие очки – принимает значения от 0 до верхнего условленного предела
  + Место – число, которое может быть от 1 до количества участников в соревновании.

Таблица 3. Описание типов данных полей

|  |  |
| --- | --- |
| Номер столбца | Описание |
| **Исходная таблица** | |
| 1 | Номер участника(уникальный ID) – тип данных integer. |
| 2 | Имя и фамилия участника(больше количество букв) – для этого поля используем массивы символов char[]. |
| 3-7 | Каждое число отвечает за количество очков, полученное на данном этапе. В связи с этим для хранения данных о всех заездах нужен массив данных типа integer. |
| 8  (опционально) | Вычисляемое поле – среднее значение по этапам. Тип данных float. |
| 9 | Для штрафов целесообразно использовать целочисленный тип данных – integer. |
| **Результирующая таблица** | |
| 1 | Имя и фамилия участника(больше количество букв) – для этого поля используем массивы символов char[]. |
| 2 | Общие очки – вычисляемое поле, однако деление не происходит, ожидаем целочисленный результат – integer. |
| 3 | Место в рейтинговой таблице – сложно вычисляемое поле, однако результат ожидаем целочисленный – используем тип данных integer. |

Так как число записей может изменятся, для увеличения возможностей программы используется динамический массив. Он позволяет варьировать количество записей, однако более сложен в реализации.

Информация, содержащаяся в файле представлена ниже:

| 2| Phil BAUHAUS| 5| 5| 5| 5| 0| 1|

| 3| Leonardo BASSO| 3| 0| 2| 3| 1| 2|

| 1| Geoffrey BOUCHARD| 1|100| 2| 3| 0| 10|

| 4| Scott DAVIES| 4| 0| 0| 1| 3| 0|

| 5| Lilian CALMEJANE| 2| 1| 0| 0| 0| 1|

Для хранения данных используем текстовый файл формата «.txt», что в отличии от бинарных файлов позволяет редактировать их, однако взамен требует чуть больше пространства. Так же он поддерживается всеми операционными системами и не требует дополнительного программного обеспечения, в отличии от табличных файлов.

В задании требуется выполнить поиск по штрафным очкам в выбираемом диапазоне. Поиск осуществляется методом сравнения (алгоритм линейного поиска). Сравнивается искомое значение со всеми значениями, если есть совпадения - результат выводится в виде списка на экран.

Согласно заданию необходимо реализовать сортировку массива по среднему значению из 5 этапов. В данной программе будет задействована быстрая сортировка – алгоритм широкого назначения С. Р. Хори, как самая быстрая и оптимизированная для значений целого типа.

Еще будет необходим вывод данных на консоль, которые хранятся в файле. Эта функция необходима для анализа имеющихся данных. А так как таблицы должно получится две, следует сделать две функции с разным набором аргументов.

В программе будут реализованы следующие функции для работы с файловой базой данных:

* Чтение – реализуется с помощью форматированного ввода из потока файла. Далее эти данные записываются в предварительно созданный список необходимого размера
* Запись - реализуется с помощью форматированного вывода в поток файла из записей списка

Также необходимо реализовать понятное и удобное меню. Оно должно быть интуитивно понятно, и иметь «изнутри» проверки вводимых значений, а также содержать такую структуру:

1 - вывод исходной таблицы

2 - вывод отсортированного по среднему значению списка

3 - вывод результирующей таблицы, содержащей очки и места

4 - поиск спортсмена по штрафным очкам в диапазоне

5 - изменение значений этапа игрока

6 - запись значений в файл

7 - добавление элемента

8 - удаление элемента

9 - выход из программы

Все задачи были поставлены. Далее начнётся конструирование самой программы, где наглядно будет показан алгоритм и ветвления программы. Алгоритм и ветвления будут проиллюстрированы с помощью блок-схем.

2 Конструирование программы

Все программы, написанные на языке СИ, обязательно содержат основную функцию, которая должна иметь имя «main». Эта функция является точкой входа в программу. Структура функции main() показана на рисунке 1.

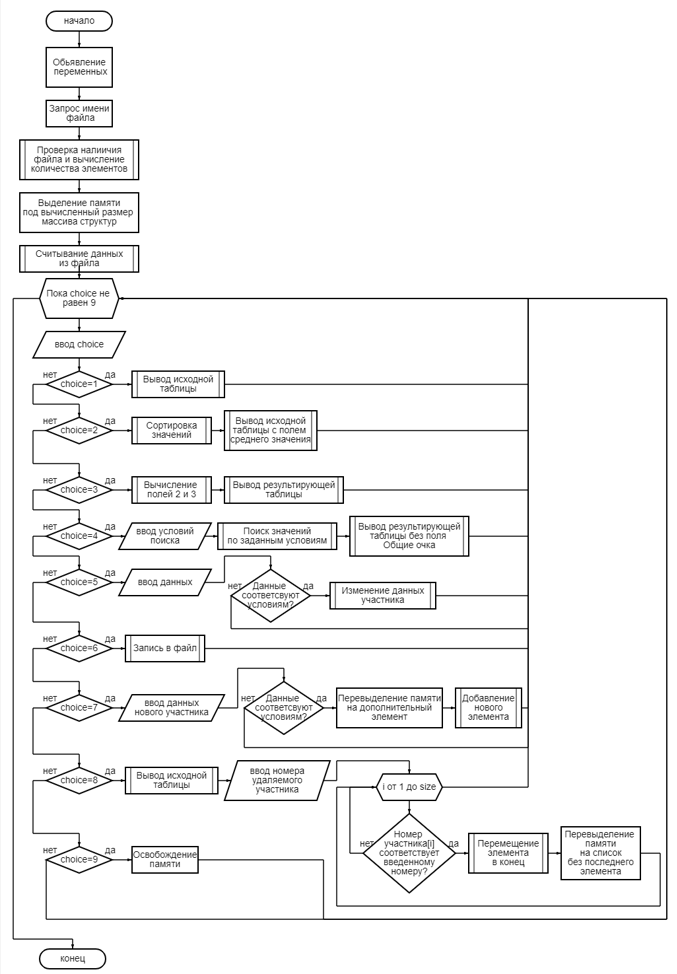


Рисунок 1 - Структура функции «main»

Краткое описание собственных функций представлено в таблице 4.

Таблица 4. Описание функций

|  |  |
| --- | --- |
| Прототип функции | Назначение |
| int check\_size(char[30]); | Служебная функция проверки размера |
| int file\_write(cyclist\*, int, char[30]); | Сохранение данных |
| int file\_read(cyclist\*, char[30]); | Считывание данных |
| void arr\_print\_data(cyclist\*, int,int); | Вывод на экран исходной таблицы |
| void arr\_print\_results(cyclist\*, int, int); | Вывод на экран результирующей таблицы |
| cyclist\* arr\_calc\_res(cyclist\*, int); | Служебная функция вычисления результатов |
| int arr\_change(cyclist\*, int, int, int, int); | Изменение значений элемента |
| cyclist\* arr\_search(cyclist\*, int, int, int, int\*); | Поиск по штрафным очкам в диапазоне |
| int add\_el(cyclist\*, int, int, char[20]); | Добавление элемента |
| int el\_swap(cyclist\*, int, int); | Обмен значениями элементов |
| int del\_el(cyclist\*, int, int); | Удаление элемента |
| int comp(cyclist\*, cyclist\*); | Служебная функция сравнения |

Функция void arr\_print\_data(cyclist\* a, int size ,int mode) печатает исходную таблицу результатов соревнования, принимаемую в списке a. Первый параметр указывает на массив структур с данными, второй параметр - size передает размер массива структур 3-й параметр устанавливает режим работы функции – для вывода различных полей. Блок-схема алгоритма представлена на рисунке 2.

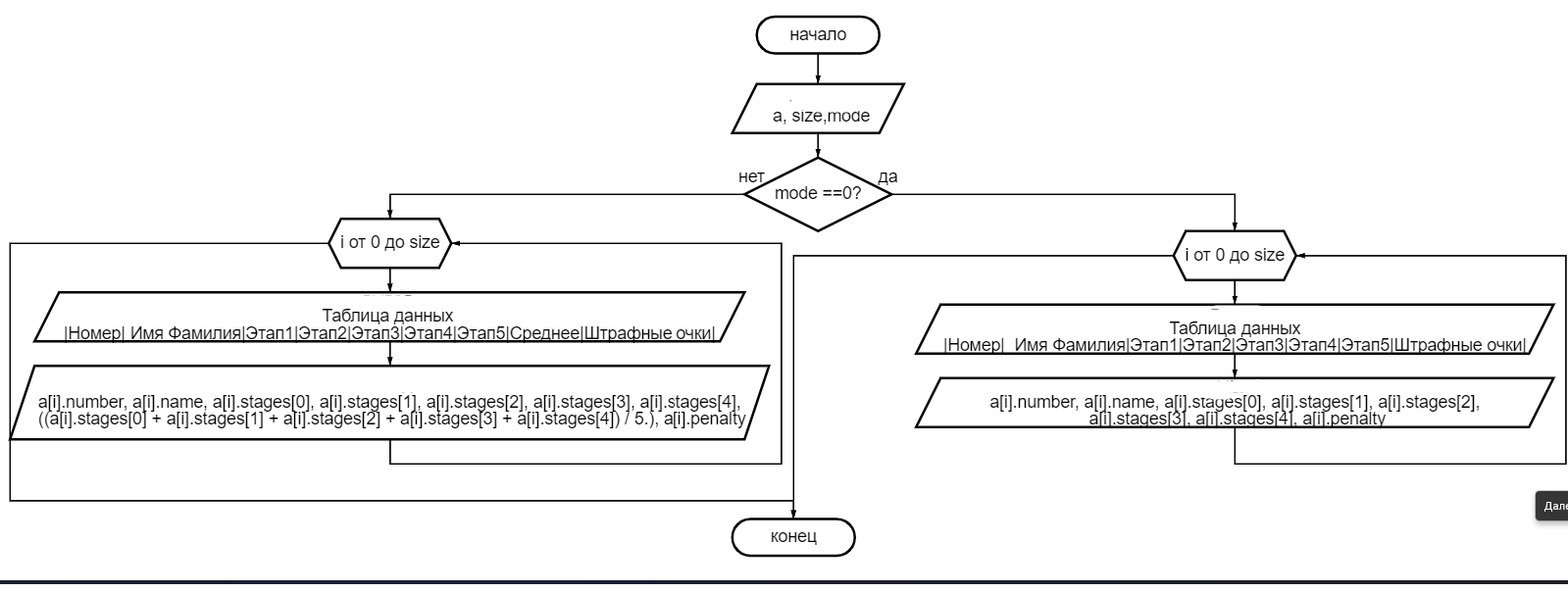


Рисунок 2 - Блок-схема функции вывода таблицы

Для сортировки используется встроенная функция void qsort(void \*base, size\_t num, size\_t size, int (\*compare) (const void \*, const void \*)), которая имеет максимальную скорость работы, используя quicksort — алгоритм сортировки широкого назначения, разработанный С. Р. Хори.

Для её работы необходима функция сравнения – compare, блок-схема которой представлена на рисунке 3.

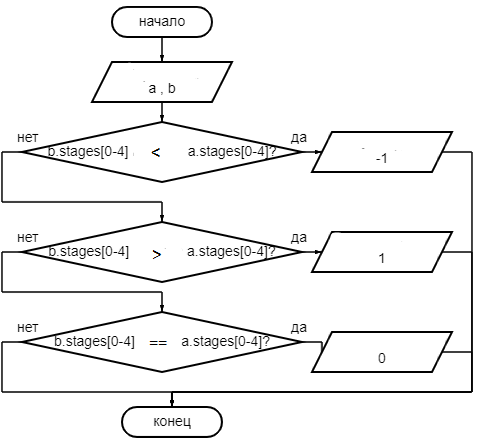


Рисунок 3 - Блок-схема функции сравнения

Функция cyclist\* arr\_calc\_res(cyclist\* a, int size) подсчитывает сумму очков участников в принимаемом списке a за 5 заездов и ранжирует их по победным местам. Параметр 1 передает указатель на массив. Параметр 2 передает размер массива. Функция возвращает указатель на подсчитанный массив. Блок-схема алгоритма представлена на рисунке 4.

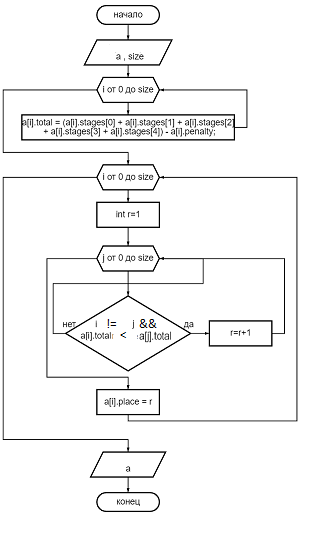


Рисунок 4 - Блок-схема функции подсчета результатов и мест

Функция cyclist\* arr\_search(cyclist\* a, int size, int lwb, int upb, int\* sz) выполняет поиск по диапазону в принимаемом списке a и возвращает указатель sz на созданный массив, который содержит найденные элементы. Параметр 1 передает ссылку на текущий массив, параметр 2 – size - передает размер массива структур, 3й и 4й параметры задают нижнюю и верхнюю границу соответственно. 5-й параметр с указателем отражает размер возвращенной выборки. Блок-схема алгоритма представлена на рисунке 5.

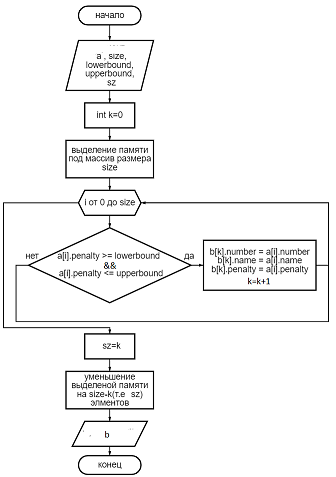


Рисунок 5 - Блок-схема функции для вывода участников по диапазону очков

Функция int arr\_change(cyclist\* a, int size, int pl\_numb, int pl\_st, int pl\_val) изменяет значение заданного этапа у заданного участника в принимаемом списке a. Параметр 1 передает указатель на массив, параметр 2 – размер массива. Параметры 3-5 передают необходимое значение игрока, этапа и значения. Возвращает 1 или 0 в зависимости от результата. Блок-схема алгоритма представлена на рисунке 6.

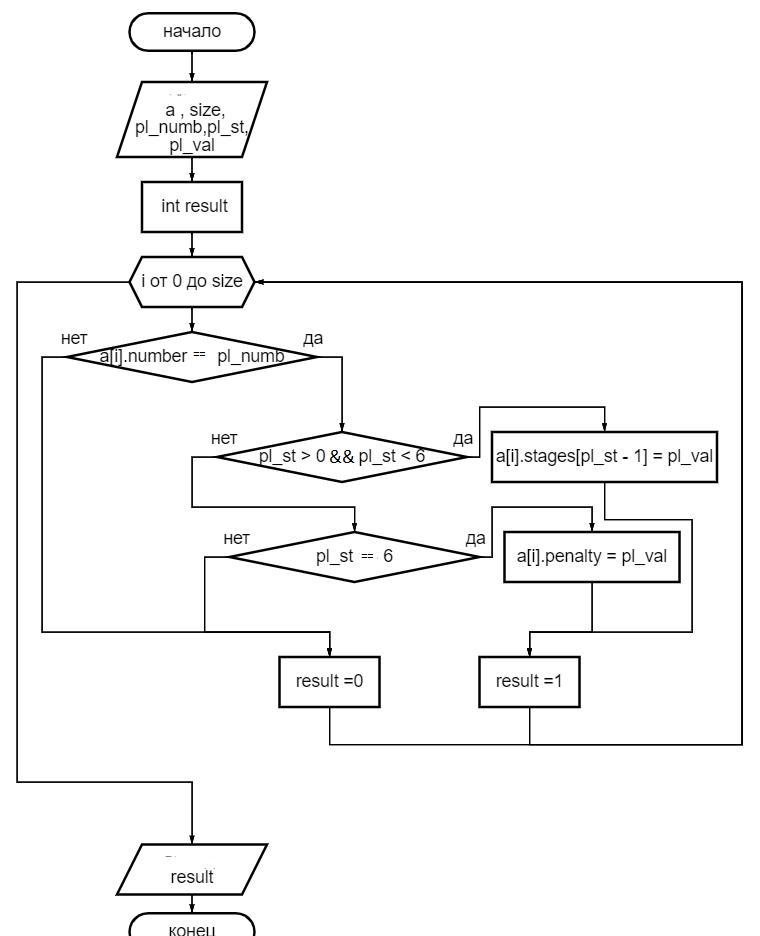


Рисунок 6 - Блок-схема функции для изменения очков.

Функция int add\_el(cyclist\* a, int size, int nmb, char nm[20]) добавляет в конец принимаемого в параметрах списка a новый элемент. Параметр 1 передает указатель на массив, параметр 2 – размер массива. Параметры 3-4 передают необходимые номер и имя игрока. Блок-схема алгоритма представлена на рисунке 7.

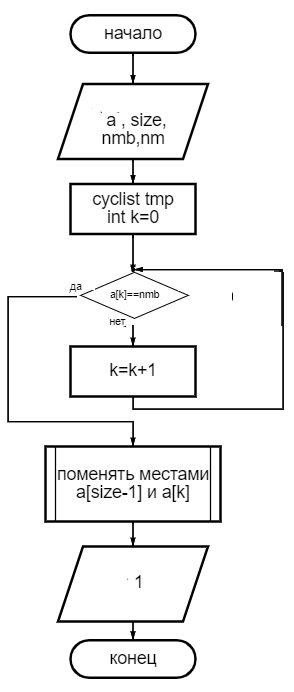


Рисунок 7 - Блок-схема функции добавления элемента

Функция int del\_el(cyclist\* a, int size, int nm) позволяет удалить элемент из принимаемого в параметрах списка a. Параметр 1 передает указатель на массив, параметр 2 – размер массива. Параметр 3 передает необходимое значение игрока. Возвращает значение 1. Блок-схема алгоритма представлена на рисунке 8.

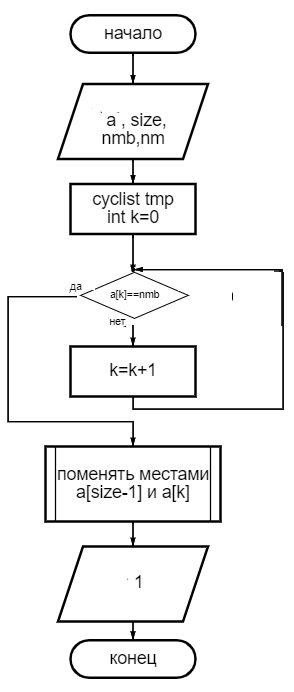


Рисунок 8 - Блок-схема функции удаления элемента.

Функция void arr\_print\_results(cyclist\* a, int size, int mode) печатает результирующую таблицу из принимаемого списка a. Первый параметр указывает на массив структур с данными, второй параметр - size размер массива структур. 3-й параметр устанавливает режим работы функции. Блок-схема алгоритма представлена на рисунке 9.

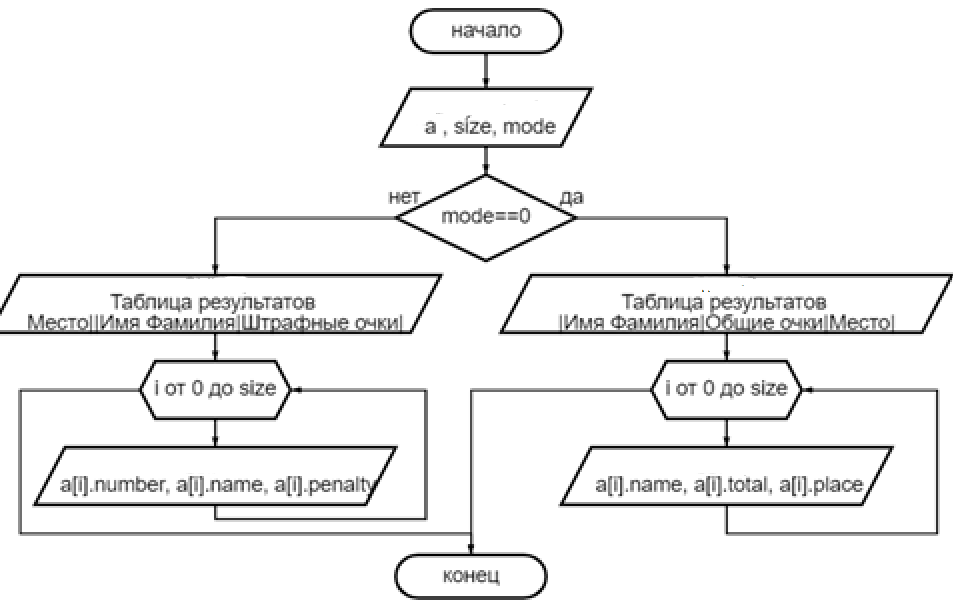


Рисунок 9 - Блок-схема функции вывода результирующей таблицы.

Функция int file\_read(cyclist\* a, char fname[30]) служит для считывания данных из файла в список a. Первый параметр указывает на массив структур с данными, Второй параметр – передает название файла. Возвращает 1. Блок-схема алгоритма представлена на рисунке 10.

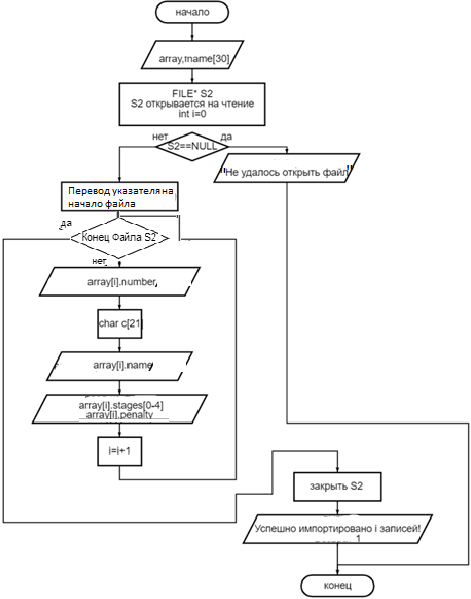


Рисунок 10 - Блок-схема функции чтения из файла.

Функция int file\_write(cyclist\* a, int size, char fname[30]) - запись данных в файл из принимаемого списка a. Первый параметр указывает на массив структур с данными, второй параметр передает размер массива, третий параметр – передает название файла. Возвращает 1. Блок-схема алгоритма представлена на рисунке 11.

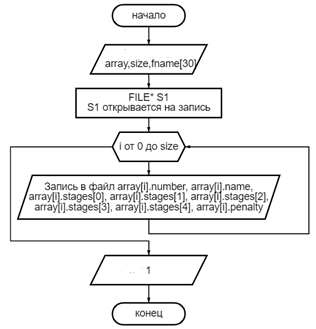


Рисунок 11 - Блок-схема функции записи в файл.

Разработаны функции: void el\_swap(cyclist\* a, int nm1, int ,nm2) – смена элементов местами в принимаемом списке a. 1-й параметр указывает на массив, 2-й и 3-й на номер элемента.

int check\_size(char fname[30]) – для проверки размера файла. принимает на вход строку с названием файла. Возвращает количество записей.

3 Тестирование программы

Полное тестирование будет проходить с данными, о которых шла речь в разделе 1 «Постановка задачи». При запуске программы выводится на экран просьба ввести имя файл из которого будут считываться данные (рисунок 12).



Рисунок 12 - Ввод имени файла, содержащего данные

Далее выводится количество данных, считанных в файле и (при каждом выборе пункта также) главное меню программы (рисунок 13).

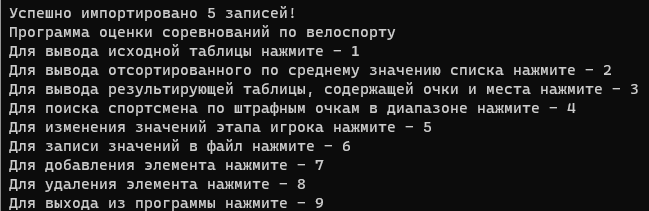


Рисунок 13 - Главное меню и результат импорта

Используя команду 1 выводится таблица исходных данных (рисунок 14).

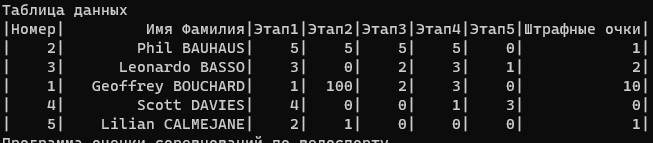


Рисунок 14 - Таблица данных

Команда 2 сортирует таблицу по среднему значению списка (рисунок 15).

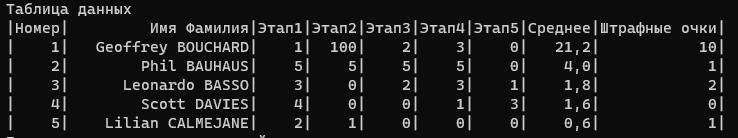


Рисунок 15 – отсортированная таблица со столбцом «среднее»

Команда 3 выводит в консоль результаты вычислений этапов, то есть результирующую таблицу (рисунок 16).

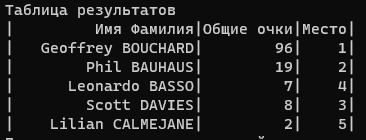


Рисунок 16 - Вывод имен яхт по диапазону очков

Команда 4 позволяет ввести нижний и верхний предел для поиска игроков по диапазону штрафных очков(рисунок 17).

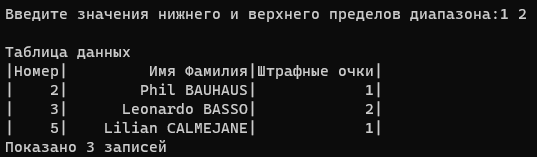


Рисунок 17 – вывод данных в диапазоне штрафных очков

Команда 5 позволяет изменить значение этапов игрока, для этого необходимо ввести данные из запроса(рисунок 18 и 19).



Рисунок 18 – Запрос данных для изменения.



Рисунок 19 – Отчет о действии.

Как видим на рисунке 20, изменения действительно внесены

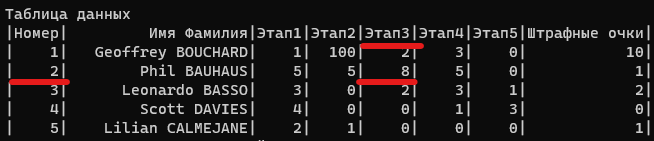


Рисунок 20 –Результат действия.

Команда 6 записывает данные в файл, и выводит соответствующую строку (рисунок 21).



Рисунок 21 –Результат действия.

Команда 7 позволяет добавить участника(рисунок 22).



Рисунок 22 - Изначальная таблица и ввод новых значений

Как видим на рисунке 23, изменения действительно внесены

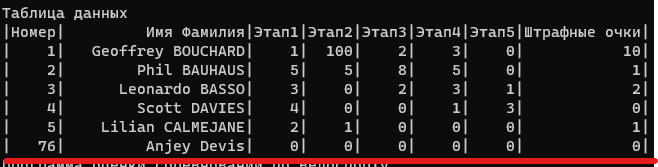


Рисунок 23 –Результат действия.

Команда 8 позволяет удалить участника(рисунок 24).

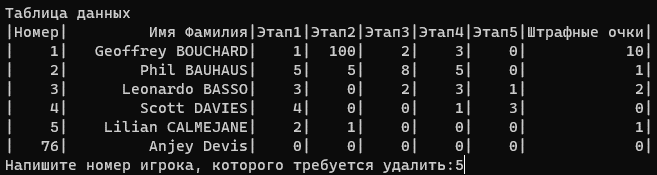


Рисунок 24 – Выбор - для удобства выбора выводится исходная таблица

После, можем убедится, что выбранное поле действительно удалено (рисунок 25)

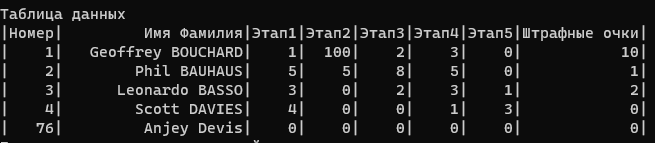


Рисунок 25 – Результат - удалено поле

Далее рассмотрим защиту от ошибки ввода пользователя

При вводе несуществующего текстового файла, выводится соответствующий фрагмент (рисунок 26)

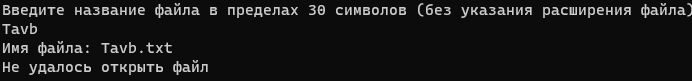


Рисунок 26 – Результат ошибки ввода

При вводе букв вместо цифр(рисунок 27) выводится соответствующий фрагмент (рисунок 28)

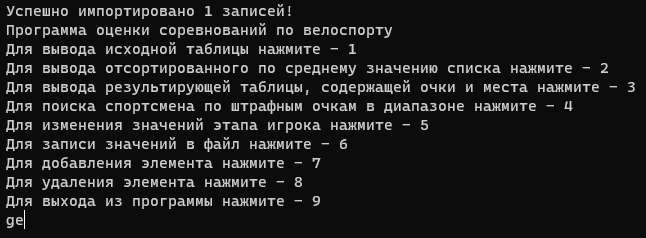


Рисунок 27 – Пример неправильного ввода



Рисунок 28 – Результат ошибки ввода

При вводе букв вместо цифр в меню №3(рисунок 29) выводится такой же фрагмент (рисунок 30)



Рисунок 29 – Пример неправильного ввода



Рисунок 30 – Результат ошибки ввода

При вводе несоответсвующих значений в меню №3 выводится пустая таблица (рисунок 31)

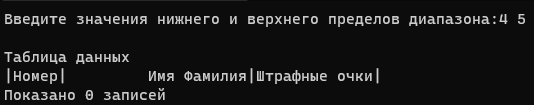


Рисунок 31 – Пример неправильного ввода

При вводе несоответсвующих значений в меню №4 выводится ответ (рисунок 32)



Рисунок 32 – Пример неправильного ввода

При вводе пустого имени(или пробела) в меню №7 выводится ответ (рисунок 33), и операция не осуществляется



Рисунок 33 – Пример неправильного ввода

ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования на языке С с использованием среды

разработки Visual Studio 2022 была реализована программа табличный процессор (Работа с таблицей).

Разработанная программа реализует следующий функционал:

* Считывание, преобразование и вывод данных из указанного пользователем файла.
* Сортировку таблицы сортировка по среднему значению из 5 этапов.
* Поиск участника по штрафным очкам в выбираемом диапазоне.
* Изменение очков этапов или штраф. очков.
* Добавление элемента в базу.
* Удаление элемента из базы.
* Выводит исходную и результирующей таблицы

Так же разработанная программа имеет ряд преимуществ:

* Динамический размер таблицы (можно изменять количество элементов вплоть до 0, а благодаря функции вычисления размера перед импортом – быть уверенным в точности выполнения и отображения данных).
* Ускоренный алгоритм сортировки(qsort), наличие работы с указателями (например, \*sz), а также переопределение памяти для «урезания» неиспользованного места, что влечет за собой увеличение скорости и уменьшение потребления памяти программой.
* Наличие проверок на вводимые значения, в том числе продвинутая обработка вводимых команд меню, не позволяющих программе зацикливаться или выполнять неправильные действия.
* Ранжирование мест участников (т.е. расчет результатов) по сложному алгоритму (описанному в постановке задачи), а также выполнение этого в отдельной функции, что позволяет использовать её многократно.
* Возможность вводить имя спортсмена с пробелом, что позволяет более широко использовать возможности текстового редактора.
* Запись и считывание данных из файла в отдельной функции, что позволяет многократно уменьшить размер кода программы и использовать его много раз

Подводя итоги, можно сказать, что все требования к программе выполнены успешно. Также, был реализован интерфейс программы, позволяющий пользователю выбирать удобным образом необходимое действие, и защищающий его от ошибок ввода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* 1. Курипта О.В. Основы программирования и алгоритмизации: практикум / О.В.Курипта, О.В. Минакова, Д.К. Проскурин; Воронежский ГАСУ. – Воронеж, 2015. – 132 с. (дата обращения: 28.12.2022)
  2. Metanit[электронный ресурс] - <https://metanit.com/> - Сайт о программировании (дата обращения: 29.12.2022)
  3. Хирьянов Тимофей[электронный ресурс] https://www.youtube.com/user/tkhirianov/ - YouTube-канал о программировании, МФТИ (дата обращения: 27.12.2022)
  4. Всероссийский киберфорум [электронный ресурс] <http://www.cyberforum.ru/>.(дата обращения: 26.12.2022)
  5. CppStudio [электронный ресурс] [http://cppstudio.com](http://cppstudio.com/). – Сайт о программировании (дата обращения: 22.12.2022)

ПРИЛОЖЕНИЕ А

Код программы

#include<stdio.h>

#include<stdlib.h>

#include<locale.h>

#include<ctype.h>

#include<string.h>

#include<math.h>

#include<stdbool.h>

#pragma warning(disable:4996)

typedef struct {

int number;

char name[21]; //для имени(20 символов и \n)

int stages[5]; //для этапов

int penalty; // для штрафных очков

int total; // расчетное поле общего результата

int place; // расчетное поле мест

} cyclist;

int check\_size(char[30]);// проверка размера файла

int file\_write(cyclist\*, int, char[30]);// запись данных в файл

int file\_read(cyclist\*, char[30]);//считывание данных из файла

void arr\_print\_data(cyclist\*, int,int);//вывод исходной таблицы

void arr\_print\_results(cyclist\*, int, int);//вывод результирующий таблицы

cyclist\* arr\_calc\_res(cyclist\*, int);//расчет результатов

int arr\_change(cyclist\*, int, int, int, int);//изменение данных этапов

cyclist\* arr\_search(cyclist\*, int, int, int, int\*);//поиск данных по диапазону

int add\_el(cyclist\*, int, int, char[20]);//добавление элемента

int el\_swap(cyclist\*, int, int);//перестановка элементов

int del\_el(cyclist\*, int, int);//удаление элементов

int comp(cyclist\*, cyclist\*);//сравнение для qsort

int main() {

setlocale(LC\_ALL, "Russian");

int size;

char extension[6] = { ".txt" };

char file\_name[30];

puts("Введите название файла в пределах 30 символов (без указания расширения файла)");

scanf("%s", file\_name);

strcat(file\_name, extension);

printf("Имя файла: %s\n", file\_name);

size = check\_size(file\_name);

cyclist\* array = (cyclist\*)malloc(size \* sizeof(cyclist));

file\_read(array, file\_name, size);

char inp;

int choice = 1;

while (choice != 9) { //главное меню

printf("Программа оценки соревнований по велоспорту\n"

"Для вывода исходной таблицы нажмите - 1\n"

"Для вывода отсортированного по среднему значению списка нажмите - 2\n"

"Для вывода результирующей таблицы, содержащей очки и места нажмите - 3\n"

"Для поиска спортсмена по штрафным очкам в диапазоне нажмите - 4\n"

"Для изменения значений этапа игрока нажмите - 5\n"

"Для записи значений в файл нажмите - 6\n"

"Для добавления элемента нажмите - 7\n"

"Для удаления элемента нажмите - 8\n"

"Для выхода из программы нажмите - 9\n"

);

scanf\_s("\n%c", &inp);//ввод команды пользователя

choice = atoi(&inp);//проверка команды на корректность

if (choice > 0 && choice <= 9) {

switch (choice) {

case 1:

system("cls");

arr\_print\_data(array, size, 0);

break;

case 2:

system("cls");

qsort(array, size, sizeof(cyclist), comp);

arr\_print\_data(array, size, 1);

break;

case 3:

system("cls");

arr\_print\_results(arr\_calc\_res(array, size), size, 0);

break;

case 4:

system("cls");

printf("Введите значения нижнего и верхнего пределов диапазона:");

int ub = 0, lb = 0;

int sz = 0;

scanf("%d", &lb);

scanf("%d", &ub);

arr\_print\_results(arr\_search(array, size, lb, ub, &sz), sz, 1);

printf("Показано %d записей\n", sz);

break;

case 5:

system("cls");

printf("Напишите номер игрока , этапа(для штрафных очков введите - 6) и новое значение:");

int pl\_numb, pl\_st, pl\_val, result;

scanf("%d %d %d", &pl\_numb, &pl\_st, &pl\_val);

(pl\_numb > 0 && pl\_st > 0 && pl\_st < 6 && pl\_val>0) ? (arr\_change(array, size, pl\_numb, pl\_st, pl\_val) == 1 ? printf("Изменение выполнено успешно!\n") : printf("Значение не найдено!\n")) : printf("Неверный ввод!");//проверка на корректность

break;

case 6:

file\_write(array, size, file\_name);

printf("Сохранено успешно!\n");

break;

case 7:

system("cls");

printf("Напишите номер нового игрока и имя:");

char pl\_name[20] = { 0 }; pl\_numb;

scanf("%d", &pl\_numb);

fgets(pl\_name, 20, stdin);

pl\_name[strlen(pl\_name) - 1] = 0;

if (pl\_numb > 0 && ((strlen(pl\_name)) > 1)) {

array = (cyclist\*)realloc(array, ++size \* sizeof(cyclist));// «добавление» памяти на 1 элемент

add\_el(array, size, pl\_numb, pl\_name);

}//проверка на корректность

else { printf("Ошибка!повторите операцию\n"); }

break;

case 8:

system("cls");

arr\_print\_data(array, size, 0);

printf("Напишите номер игрока, которого требуется удалить:");

scanf("%d", &pl\_numb);

for (int i = 0; i < size; i++) {//проверка на корректность

if (array[i].number == pl\_numb) {

size > 0 && pl\_numb > 0 ? del\_el(array, size, pl\_numb) : printf("Размер массива - 0, удалять нечего!\n");

size > 0 && pl\_numb > 0 ? array = (cyclist\*)realloc(array, --size \* sizeof(cyclist)) : printf("Ошибка!повторите операцию\n");

}

}

system("cls");

break;

}

}

else {

system("cls");

printf("Ошибка! Повторите ввод!\n");

}

};

free(array);//освобождение памяти

exit(0);//безопасный выход

}

int check\_size(char fname[30])

{

FILE\* S2 = fopen(fname, "rt");

if (S2 == NULL) {

puts("Не удалось открыть файл");

exit(0);

}

int size;

int k = 0;

while (!feof(S2)) {

if (fgetc(S2) == '\n') {

k++;

}

}

size = k;

fclose(S2); //освобождение памяти

return size;

}

int file\_read(cyclist\* array, char fname[30])

{

system("cls");

FILE\* S2 = fopen(fname, "rt");

if (S2 == NULL) {

puts("Не удалось открыть файл");

exit(0); //безопасный выход

}

int i = 0;

fseek(S2, 0, SEEK\_SET);

while (!feof(S2)) {

fscanf(S2, "|%5d|", &(array + i)->number);

char c[21];

fgets(c, 21, S2);//отдельный ввод для сохранения пробелов

strcpy(&array[i].name, c);

fscanf(S2, "|%5d|%5d|%5d|%5d|%5d|%13d|\n", &(array + i)->stages[0], &(array + i)->stages[1], &(array + i)->stages[2], &(array + i)->stages[3], &(array + i)->stages[4], &(array + i)->penalty);

i++;

}

fclose(S2); //освобождение памяти

printf("Успешно импортировано %d записей!\n", i);

return i;

}

int file\_write(cyclist\* array, int size, char fname[30])

{

//запись в файл

system("cls");

FILE\* S1 = fopen(fname, "w");

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

fprintf(S1, "|%5d|%20s|%5d|%5d|%5d|%5d|%5d|%13d|\n", array[i].number, array[i].name, array[i].stages[0], array[i].stages[1], array[i].stages[2], array[i].stages[3], array[i].stages[4], array[i].penalty);

};

fclose(S1); //освобождение памяти

return 1;

}

void arr\_print\_data(cyclist\* a, int size, int mode)

{

if (mode == 0) {//выбор режима

printf("\nТаблица данных\n");

printf("|%2s|%20s|%5s|%5s|%5s|%5s|%5s|%5s|\n", "Номер", "Имя Фамилия", "Этап1", "Этап2", "Этап3", "Этап4", "Этап5", "Штрафные очки");

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

printf("|%5d|%20s|%5d|%5d|%5d|%5d|%5d|%13d|\n", a[i].number, a[i].name, a[i].stages[0], a[i].stages[1], a[i].stages[2], a[i].stages[3], a[i].stages[4], a[i].penalty);

};

}

else {

printf("\nТаблица данных\n");

printf("|%2s|%20s|%5s|%5s|%5s|%5s|%5s|%7s|%5s|\n", "Номер", "Имя Фамилия", "Этап1", "Этап2", "Этап3", "Этап4", "Этап5", "Среднее", "Штрафные очки");

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

printf("|%5d|%20s|%5d|%5d|%5d|%5d|%5d|%7.1f|%13d|\n", a[i].number, a[i].name, a[i].stages[0], a[i].stages[1], a[i].stages[2], a[i].stages[3], a[i].stages[4], ((a[i].stages[0] + a[i].stages[1] + a[i].stages[2] + a[i].stages[3] + a[i].stages[4]) / 5.), a[i].penalty);

};

}

}

void arr\_print\_results(cyclist\* a, int size, int mode)

{

if (mode == 0) {//выбор режима

printf("\nТаблица результатов\n");

printf("|%20s|%5s|%5s|\n", "Имя Фамилия", "Общие очки", "Место");

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

printf("|%20s|%10d|%5d|\n", a[i].name, a[i].total, a[i].place);

};

}

else {

printf("\nТаблица данных\n");

printf("|%2s|%20s|%5s|\n", "Номер", "Имя Фамилия", "Штрафные очки");

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

printf("|%5d|%20s|%13d|\n", a[i].number, a[i].name, a[i].penalty);

}

}

}

cyclist\* arr\_calc\_res(cyclist\* a, int size)

{

for (int i = 0; i < size; i++)//вычисление общих очков

a[i].total = (a[i].stages[0] + a[i].stages[1] + a[i].stages[2] + a[i].stages[3] + a[i].stages[4]) - a[i].penalty;

for (int i = 0; i < size; i++) {//вычисление мест игроков

int r = 1;

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

if (i != j && a[i].total < a[j].total)

r += 1;

}

a[i].place = r;

}

return a;

}

cyclist\* arr\_search(cyclist\* a, int size, int lowerbound, int upperbound, int\* sz)

{

int k = 0;

cyclist\* b = (cyclist\*)malloc(size \* sizeof(cyclist));

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

if (a[i].penalty >= lowerbound && a[i].penalty <= upperbound) {

b[k].number = a[i].number;

strcpy(b[k].name, a[i].name);

b[k].penalty = a[i].penalty;

k++;

}

}

\*sz = k;

b = (cyclist\*)realloc(b, \*sz \* sizeof(cyclist));// «удаление» пустых данных

return b;

}

int arr\_change(cyclist\* a, int size, int pl\_numb, int pl\_st, int pl\_val)

{

int result;

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

if (a[i].number == pl\_numb) { //проверка на корректность

pl\_st > 0 && pl\_st < 6 ? a[i].stages[pl\_st - 1] = pl\_val : false;

pl\_st == 6 ? a[i].penalty = pl\_val : false;

result = 1;

break;

}

else {

result = 0;

}

}

return result;

}

int add\_el(cyclist\* a, int size, int nmb, char nm[20]) {

strncpy(a[size - 1].name, nm, 20);

(a + (size - 1))->number = nmb;

for (int i = 0; i < 5; i++) {

a[size - 1].stages[i] = 0;

}

a[size - 1].penalty = 0;

return 1;

};

int el\_swap(cyclist\* a, int ind1, int ind2) {

cyclist tmp;

tmp = a[ind1];

a[ind1] = a[ind2];

a[ind2] = tmp;

}

int del\_el(cyclist\* a, int size, int nmb) {

cyclist tmp; int k = 0;

while (a[k].number != nmb)k++;

el\_swap(a, size - 1, k);

return 1;

};

int comp(cyclist\* a, cyclist\* b)

{

if (((b->stages[0] + b->stages[1] + b->stages[2] + b->stages[3] + b->stages[4]) / 5.) < ((a->stages[0] + a->stages[1] + a->stages[2] + a->stages[3] + a->stages[4]) / 5.)) {

return -1;

}

else if (((b->stages[0] + b->stages[1] + b->stages[2] + b->stages[3] + b->stages[4]) / 5.) > ((a->stages[0] + a->stages[1] + a->stages[2] + a->stages[3] + a->stages[4]) / 5.)) {

return 1;

}

else if (((b->stages[0] + b->stages[1] + b->stages[2] + b->stages[3] + b->stages[4]) / 5.) == ((a->stages[0] + a->stages[1] + a->stages[2] + a->stages[3] + a->stages[4]) / 5.)) {

return 0;

}

};

// ссылка на github: https://github.com/thewildcat19/Curse-w2