

Trabalho: TP1

Nome: Vitor Cláudio Chaves de Aguiar

Matrícula: 2015053292

Curso: Sistemas de Informação

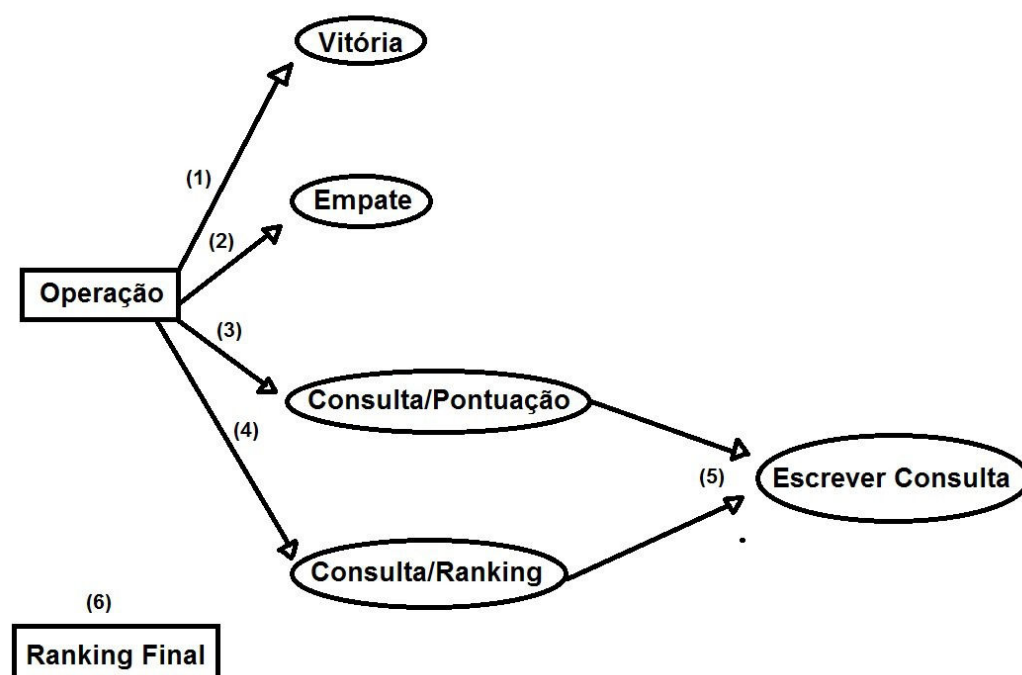
## 1. Introdução

É possível observar que o mundo é repleto de esportes e por sua vez possui diversas competições. O futebol, muito famoso no Brasil, é repleto de campeonatos. Nos campeonatos, as informações de pontuação ou ranking dos times é desejada por jornalistas e fãs do futebol.

O objetivo desse trabalho é possibilitar a consulta por pontuação e por ranking do campeonato pelo usuário. Desta forma todas as pessoas podem acompanhar os resultados em tempo real de uma forma eficiente.

## 2. Implementação

Fluxograma do programa



1- Soma 3 pontos para o vencedor da partida.

2- Soma 1 ponto para os dois times.

3- Retorna a pontuação do time consultado.

4- Retorna o ranking do time consultado.

5- Escreve no arquivo a consulta.

6- Escreve no arquivo o ranking no final de todas as operações.

## Estrutura de dados

Para a implementação do trabalho foram usados 2 vetores que armazenam elementos do tipo Time. Esses vetores foram alocados com  $2 * \text{número de linhas} + 1$ , pois este é o valor máximo de times distintos possíveis com determinado número de linhas. (O +1 foi utilizado pois a posição 0 dos vetores é usada como sentinela).

O único struct criado foi o struct Time. Sendo esta estrutura capaz de armazenar um nome e uma pontuação. A informação do ranking é obtida através do índice do vetor que está ordenado pela pontuação.

## Funções e Procedimentos

### TAD Arquivo

**char \*strtok\_single (char \* str, char const \* delims):** recebe a string e o separador. Diferente do strtok convencional, a função separa a string quando encontra o separador porém não ocorre problemas ao se deparar com dois separadores seguidos.

**int leQuantidadeDeLinhas(char \*caminhoDoArquivo, Time \*\*times):** recebe o caminho do arquivo e um vetor de cliente. Abre o arquivo para ler e contar quantas linhas ele possui com a finalidade de alocar memória para o vetor de clientes.

**void verificaOperacao(char \*operacao, Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, int \*numeroDeTimes, FILE \*saida):** Verifica se é uma operação de vitória/empate ou consulta.

**void analisaLinhaVitoriaOuEmpate(char \*operacao, Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, int \*numeroDeTimes):** Analisa os dados da linha quando a operação é vitória ou empate.

**void analisaLinhaConsulta(Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, int \*numeroDeTimes, FILE \*saida):** Analisa os dados da linha quando a operação é consulta.

**void analisaArquivo(char \*caminhoDoArquivo, Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, FILE \*saida):** Faz validações do arquivo de entrada e lê linha por linha para fazer as operações desejadas.

### TAD Time

**int buscaTimeNome(Time \*times, char \*nome, int \*tamanhoVetor):** Faz uma pesquisa binária sendo que a chave buscada é o nome do time.

**int buscaTimePontuacao(Time \*times, int pontuacao, int \*tamanhoVetor):** Faz uma pesquisa binária sendo que a chave buscada é a pontuação do time.

**void insereVetorTimeNome(Time \*times, char \*nome, int \*final):** Insere na última posição do vetor de times ordenados por nome.

**void insereVetorTimePontuacao(Time \*times, char \*nome, int \*final):** Insere na última posição do vetor de times ordenados por pontuação.

**void insereTimeQueNaoJogou(Time \*times, char \*nome, int \*final):** Insere na última posição do vetor de times que não jogaram.

**void OrdenaPorPontuacao(Time \*times, int \*n):** Ordena o vetor pela chave pontuação e considerando a ordem alfabética no empate utilizando o método de Inserção.

**void OrdenaPorNome(Time \*times, int \*n):** Ordena o vetor pela chave nome utilizando o método de Inserção.

**void fazOperacaoVitoria(Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, int indiceVetor, int indiceVetor2):** Soma 3 pontos em determinado time nos vetores timesOrdenadoNome e timesOrdenadoPontuacao.

**void fazOperacaoEmpate(Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, int indiceVetor, int indiceVetor2):** Soma 1 ponto para dois times nos vetores timesOrdenadoNome e timesOrdenadoPontuacao.

**void fazConsultaRanking(char \*nome, int ranking, FILE \*saida):** Procura time e retorna a sua posição no ranking.

**void fazConsultaPontuacao(char \*nome, int pontuacao, FILE \*saida):** Procura time e retorna a sua pontuação.

**void fazListaRanking(Time \*timesOrdenadoPontuacao, FILE \*saida, int \*numeroDeTimes):** Retorna o ranking final do campeonato, após todas as operações.

### Programa Principal

O programa principal valida se os argumentos passados na execução do programa estão corretos. Cria os dois vetores de times que serão utilizados no programa. 1 ordenado pelo nome e 1 ordenado por pontuação.

### Organização do Código, Decisões de Implementação e Detalhes Técnicos

O código está dividido em 2 arquivos .h e 3 arquivos .c, sendo o arquivo Main.c o programa principal. Foi utilizado o compilador gcc no Ubuntu e o valgrind com os parâmetros `-leak-check=full`.

No Arquivo.c o arquivo de entrada é lido linha por linha e se verifica a operação que vai ser feita. Cada operação de vitória/empate, que são operações que mudam os valores de pontuação dos times, posteriormente é ordenado os vetores novamente. Ou seja, os vetores sempre estão ordenados pelas suas respectivas chaves. Quando a operação é consulta por pontuação, retorna 0 pontos se o time não jogou e se o time jogou uma pesquisa binária é feita no vetor ordenado por nome e pega a pontuação desse time encontrado. Quando a operação é consulta por ranking, retorna n+1 se o time não jogou e se o time jogou faz uma pesquisa binária no vetor ordenado por nome, pega a pontuação desse time, faz outra pesquisa binária analisando as ocorrências dessa pontuação no vetor ordenado por pontuação e verificando qual delas possui o nome do time desejado.

Inserção nos vetores: se insere um time na última posição do vetor.

Ordenação dos vetores: é utilizado o método de inserção (Insertion Sort), pois o vetor se mantém ordenado o tempo todo e a alteração que é feita nele é mínima. Então o

desempenho desse método em um vetor quase ordenado é superior que os outros métodos.

Busca nos vetores: é utilizado a pesquisa binária que possui complexidade  $O(\log n)$ .

### 3. Análise de complexidade

#### TAD Arquivo

**char \*strtok\_single (char \* str, char const \* delims)**: executa apenas comandos  $O(1)$ . Portanto temos a complexidade  $O(1)$ .

**int leQuantidadeDeLinhas(char \*caminhoDoArquivo, Time \*\*times)**: possui um loop que depende de quando o arquivo vai terminar, ou seja, vai executar  $n$  vezes o loop. Portanto temos a complexidade  $O(n)$ .

**void verificaOperacao(char \*operacao, Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, Time \*timesQueNaoJogaram, int \*numeroDeTimes, FILE \*saida, int \*tamanhoVetorNaoJogaram)**: executa apenas comandos  $O(1)$ . Portanto temos a complexidade  $O(1)$ .

**void analisaLinhaVitoriaOuEmpate(char \*operacao, Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, Time \*timesQueNaoJogaram, int \*numeroDeTimes, int \*tamanhoVetorNaoJogaram)**: as inserções realizadas são  $O(1)$ , as buscas realizadas são  $O(\log n)$ , as ordenações realizadas são  $O(n \log n)$ . Portanto temos a complexidade  $O(\log n)$ .

**void analisaLinhaConsulta(Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, Time \*timesQueNaoJogaram, int \*numeroDeTimes, FILE \*saida, int \*tamanhoVetorNaoJogaram)**: as inserções realizadas são  $O(1)$ , as buscas realizadas são  $O(\log n)$ , as ordenações realizadas são  $O(n \log n)$ , a escrita do resultado no arquivo é  $O(n)$ . Portanto temos a complexidade  $O(\log n)$ .

**void analisaArquivo(char \*caminhoDoArquivo, Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, Time \*timesQueNaoJogaram, FILE \*saida)**: possui um loop que verifica linha por linha do arquivo, ou seja,  $O(n)$ . Porém as funções listadas acima são executadas nele, portanto fica um while dentro de um while gerando a complexidade  $O(n^2)$ . Sendo assim, temos a complexidade  $O(n^2)$ .

#### TAD Time

**int buscaTimeNome(Time \*times, char \*nome, int \*tamanhoVetor)**: é uma pesquisa binária, portanto temos a complexidade  $O(\log n)$ .

**int buscaTimePontuacao(Time \*times, int pontuacao, int \*tamanhoVetor)**: é uma pesquisa binária, portanto temos a complexidade  $O(\log n)$ .

**void insereVetorTimeNome(Time \*times, char \*nome, int \*final)**: executa apenas comandos  $O(1)$ . Portanto temos a complexidade  $O(1)$ .

**void insereVetorTimePontuacao(Time \*times, char \*nome, int \*final)**: executa apenas comandos  $O(1)$ . Portanto temos a complexidade  $O(1)$ .

**void insereTimeQueNaoJogou(Time \*times, char \*nome, int \*final):** executa apenas comandos  $O(1)$ . Portanto temos a complexidade  $O(1)$ .

**void OrdenaPorPontuacao(Time \*times, int \*n):** ordena pelo método de Inserção, portanto possui complexidade  $O(n)$ .

**void OrdenaPorNome(Time \*times, int \*n):** ordena pelo método de Inserção, portanto possui complexidade  $O(n)$ .

**void fazOperacaoVitoria(Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, int indiceVetor, int indiceVetor2):** executa apenas comandos  $O(1)$ . Portanto temos a complexidade  $O(1)$ .

**void fazOperacaoEmpate(Time \*timesOrdenadoNome, Time \*timesOrdenadoPontuacao, int indiceVetor, int indiceVetor2):** executa apenas comandos  $O(1)$ . Portanto temos a complexidade  $O(1)$ .

**void fazConsultaRanking(char \*nome, int ranking, FILE \*saida):** executa apenas comandos  $O(1)$ . Portanto temos a complexidade  $O(1)$ .

**void fazConsultaPontuacao(char \*nome, int pontuacao, FILE \*saida):** executa apenas comandos  $O(1)$ . Portanto temos a complexidade  $O(1)$ .

**void fazListaRanking(Time \*timesOrdenadoPontuacao, Time \*timesQueNaoJogaram, FILE \*saida, int \*numeroDeTimes, int \*tamanhoVetorNaoJogaram):** imprime o vetor de pontuação, ou seja, percorre o vetor inteiro em um loop. Sendo assim, temos a complexidade  $O(n)$ .

**PROGRAMA PRINCIPAL:** O programa vai assumir a complexidade da função analisaArquivo que é a primeira função a ser chamada e todas as operações são realizadas nela. Ou seja, temos a complexidade  $O(n^2)$ .

**ANÁLISE EM FUNÇÃO DO NÚMERO DE TIMES:** Quanto maior for o número de times não repetidos nas linhas do arquivo, maior serão os vetores que os possuem. Quando isso pode afetar negativamente:


- Quando for inserir resultados no arquivo sendo que é percorrido o vetor todo para fazer isso.
- Quando for ordenar ou buscar em um vetor maior.

**ANÁLISE EM FUNÇÃO DO NÚMERO DE CONSULTAS:** Se o arquivo de entrada possuir mais operações de consulta o tempo de execução será menor pois:

- Não acontece nenhuma inserção.
- Não acontece nenhuma ordenação.
- A consulta tem complexidade  $O(\log n)$ , que é muito bom.

## 4. Testes

### 4.1 Passagem de argumento incorreta na execução do programa:

 vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds

```
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ gcc -g -Wall -std=c99 *.c -o tp2
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ ./tp2 arquivo.csv

    O numero de argumentos esta errado. O programa deve receber 2 argumentos.

vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ █
```


### 4.2 Quando alguma linha do arquivo de entrada possui tamanho maior que 138 caracteres:

```
vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ gcc -g -Wall -std=c99 *.c -o tp2
vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ ./tp2 arquivo.csv arquivo2.csv

    Numero de caracteres na linha invalido. Maximo eh 138.

vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ █
```

### 4.3 Quando alguma operação do arquivo de entrada é diferente de VITORIA, EMPATE ou CONSULTA :

 vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds

```
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ gcc -g -Wall -std=c99 *.c -o tp2
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ ./tp2 arquivo.csv arquivo2.csv

    Alguma operacao de um cliente no arquivo esta invalida.

vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ █
```

### 4.4 Quando em alguma linha do arquivo de entrada falta ou sobra alguma informação (possui algo diferente do que 2 vírgulas):

```
vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ gcc -g -Wall -std=c99 *.c -o tp2
vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ ./tp2 arquivo.csv arquivo2.csv

    Numero de informacoes em uma linha esta incorreto.

vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ █
```

### 4.5 Quando algum campo estiver vazio ou a linha estiver vazia:

```
vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ gcc -g -Wall -std=c99 *.c -o tp2
vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ ./tp2 arquivo.csv arquivo2.csv

    Numero de informacoes em uma linha esta incorreto.

vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ █
```

#### 4.6 Quando o nome do time tiver mais do que 63 caracteres:

```
vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ gcc -g -Wall -std=c99 *.c -o tp2
vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$ ./tp2 arquivo.csv arquivo2.csv

Tamanho de um nome de time esta incorreto. Maximo de 63 caracteres.

vitor-aguiar@vitoraguiar-desktop:~/Área de Trabalho/C/TP2$
```

#### 4.7 Quando a operação de consulta é diferente de PONTUACAO ou RANKING:

```
vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ gcc -g -Wall -std=c99 *.c -o tp2
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ ./tp2 arquivo.csv arquivo2.csv

Tipo de consulta invalida.

vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$
```

#### 4.8 Quando consultar por pontuação time que não jogou ainda:

arquivo.csv	arquivo2.csv
1 CONSULTA,PONTUACAO,LEOES LEAIS	1 PONTUACAO,LEOES LEAIS,0
	2

#### 4.9 Quando consultar por ranking time que não jogou ainda:

arquivo.csv	arquivo2.csv
1 CONSULTA,RANKING,LEOES LEAIS	1 RANKING,LEOES LEAIS,1
	2

#### 4.10 Arquivo de entrada com apenas operação de VITORIA e EMPATE:

arquivo.csv	arquivo2.csv
1 VITORIA,LEOES LEAIS,POMBOS POTENTES	1 1,JACUS JUSTOS,12
2 EMPATE,LEOES LEAIS,FOCAS FORTES	2 2,LEOES LEAIS,6
3 VITORIA,POMBOS POTENTES,JACUS JUSTOS	3 3,FOCAS FORTES,3
4 VITORIA,JACUS JUSTOS,POMBOS POTENTES	4 4,POMBOS POTENTES,3
5 EMPATE,LEOES LEAIS,FOCAS FORTES	5
6 EMPATE,LEOES LEAIS,FOCAS FORTES	
7 VITORIA,JACUS JUSTOS,POMBOS POTENTES	
8 VITORIA,JACUS JUSTOS,POMBOS POTENTES	
9 VITORIA,JACUS JUSTOS,POMBOS POTENTES	



#### 4.11 Arquivo de entrada apenas com operação de CONSULTA:

arquivo.csv	arquivo2.csv
1 CONSULTA,RANKING,LEOES LEAIS	1 RANKING,LEOES LEAIS,1
2 CONSULTA,PONTUACAO,POMBOS POTENTES	2 PONTUACAO,POMBOS POTENTES,0
3 CONSULTA,PONTUACAO,FOCAS FORTES	3 PONTUACAO,FOCAS FORTES,0
4 CONSULTA,RANKING,FOCAS FORTES	4 RANKING,FOCAS FORTES,1
5 CONSULTA,RANKING,TIGRES TENAZES	5 RANKING,TIGRES TENAZES,1
6 CONSULTA,RANKING,LEOES LEAIS	6 RANKING,LEOES LEAIS,1
7 CONSULTA,RANKING,JACUS JUSTOS	7 RANKING,JACUS JUSTOS,1
8 CONSULTA,PONTUACAO,IGUANAS IMBATIVEIS	8 PONTUACAO,IGUANAS IMBATIVEIS,0
9 CONSULTA,RANKING,TIGRES TENAZES	9 RANKING,TIGRES TENAZES,1
10 CONSULTA,RANKING,LEOES LEAIS	10 RANKING,LEOES LEAIS,1
11 CONSULTA,RANKING,JACUS JUSTOS	11 RANKING,JACUS JUSTOS,1
12 CONSULTA,PONTUACAO,IGUANAS IMBATIVEIS	12 PONTUACAO,IGUANAS IMBATIVEIS,0
	13

#### 4.12 Arquivo de entrada com muitas linhas:

arquivo.csv	arquivo2.csv
1 VITORIA,LEOES LEAIS,POMBOS POTENTES	1 RANKING,LEOES LEAIS,1
2 CONSULTA,RANKING,LEOES LEAIS	2 PONTUACAO,POMBOS POTENTES,0
3 CONSULTA,PONTUACAO,POMBOS POTENTES	3 PONTUACAO,FOCAS FORTES,0
4 CONSULTA,PONTUACAO,FOCAS FORTES	4 RANKING,FOCAS FORTES,3
5 VITORIA,JACARE BANGUELA,FOCAS FORTES	5 RANKING,TIGRES TENAZES,5
6 CONSULTA,RANKING,FOCAS FORTES	6 RANKING,JACARE BANGUELA,2
7 CONSULTA,RANKING,TIGRES TENAZES	7 RANKING,JACUS JUSTOS,5
8 EMPATE,LEOES LEAIS,FOCAS FORTES	8 PONTUACAO,IGUANAS IMBATIVEIS,0
9 VITORIA,POMBOS POTENTES,JACUS JUSTOS	9 1,JACARE BANGUELA,9
10 CONSULTA,RANKING,JACARE BANGUELA	10 2,LEOES LEAIS,4
11 VITORIA,JACARE BANGUELA,FOCAS FORTES	11 3,JACUS JUSTOS,3
12 CONSULTA,RANKING,JACUS JUSTOS	12 4,POMBOS POTENTES,3
13 CONSULTA,PONTUACAO,IGUANAS IMBATIVEIS	13 5,FOCAS FORTES,2
14 EMPATE,TUBARAO FORTE,FOCAS FORTES	14 6,TUBARAO FORTE,1
15 VITORIA,JACUS JUSTOS,POMBOS POTENTES	15
16 VITORIA,JACARE BANGUELA,TUBARAO FORTE	

#### 4.13 Arquivo de entrada com 1 linha:

arquivo.csv	arquivo2.csv
1 VITORIA,LEOES LEAIS,POMBOS POTENTES	1 1,LEOES LEAIS,3
	2 2,POMBOS POTENTES,0
	3

#### 4.14 Quando campo de nome está vazio:

```
vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ ./tp2 arquivo.csv arquivo2.csv

Campo nome esta vazio.

vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$
```



## 5. Conclusão

É possível analisar neste trabalho como os métodos de ordenação e de pesquisa podem ser utilizados em uma situação real. O quanto uma pesquisa ou uma ordenação podem ser executadas muitas vezes e, diante disso, cria-se a necessidade de implementar uma solução que não custará muito para o programa devolver as saídas necessárias.

A maior dificuldade encontrada foi como ordenar o vetor de pontuação com o critério de desempate pela ordem alfabética do nome.

## 6. Referências

- <http://stackoverflow.com/>
- <http://www.cplusplus.com/>
- <http://www.cprogressivo.net/>
- <http://www.tutorialspoint.com/>

## 7. Anexos

Listagem dos programas:

- Time.c
- Time.h
- Arquivo.c
- Arquivo.h
- Main.c