

Copa do Mundo

Caio Matheus, Christian Luis, Alex S. Lacerda, Fillype Nascimento,
José Fortes, Rodrigo L. Rocha, Natalia O. Borges

Prof. Dr. Marcos F. Caetano
CIC116319 Estruturas de Dados, Turma A.
Departamento de Ciência da Computação
Universidade de Brasília

23 de Maio de 2018

1 Introdução



A corrida pela Taça do Mundo está acirrando-se cada vez mais. Cada time treinou bastante para chegar aonde chegou e nenhum desistirá de ser o campeão do mundo. **Escolha seu time!** Mas a vitória ainda está longe. Sempre há mais alguma partida a ser jogada e o seu papel será implementar

uma Copa do Mundo. Com times pré-definidos, você terá que gerar partidas e vence-las para se tornar o grande campeão mundial!

2 Objetivos

A representação das partidas será feita utilizando a estrutura de dados do tipo árvore binária. Nesta estrutura, cada nó representará um time. A partida acontecerá através da comparação entre as habilidades de dois times, sendo o vencedor aquele que tiver o maior valor para a habilidade escolhida. O vencedor da Copa do Mundo será aquele que conseguir vencer todas as partidas e se tornar, enfim, o campeão mundial.

2.1 Árvore Binária - Copa do Mundo

A ordem das partidas que fazem parte da Copa é definida pela árvore do tipo binária, formada por 5 níveis. Nesta estrutura, inicialmente, os galhos representam a ordem em que as partidas serão realizadas e os nós do tipo folha representam as seleções. Neste caso, teremos 16 times diferentes que participarão da Copa. Ao final das partidas, o nó raiz representará o grande Campeão Mundial.

Cada nó da árvore possuirá uma estrutura do tipo **Team *** (vide Seção 2.2). No início do jogo, apenas os nós-folha estarão preenchidos com os times que participarão da Copa. As partidas serão realizadas entre 2 nós filhos de um mesmo nó-pai. O vencedor de uma partida passa a ocupar a posição de seu nó-pai e "sobe" na árvore para a próxima etapa, onde enfrentará o outro nó-filho de mesmo pai. Este procedimento será repetido até que o vencedor ocupe a posição do nó-raíz.

Todos os times começarão no nível 4 da árvore (*Oitavas de Final*). Os vencedores das Oitavas de Final serão promovidos para o nível 3 da árvore (*Quartas de Final*). Os vencedores das Quartas de Final irão para o nível 2 da árvore (*Semi-Finais*). Os vencedores das Semi-finais irão para o nível 1 da árvore (*Final*). Finalmente, o vencedor da Final irá para o nível 0 da árvore (*Campeão*) e vence a Copa do Mundo.

2.2 Times/Seleções

A árvore binária que representa a fase de mata-mata da Copa do Mundo é formada pela estrutura do tipo **t_node**, conforme apresentado pela Figura 1. Cada elemento **t_node**, é formado por três ponteiros. Os ponteiros **left** e

`right` contém endereços para os elementos da esquerda e direita, respectivamente, caso eles existam. Se o nó representado for do tipo folha, os ponteiros `left` e `right` estarão apontados para `NULL`. O ponteiro `team` contém o endereço para estrutura do tipo `Team`, conforme definido pela Figura 2. Caso um nó-galho ainda não armazene uma Seleção, o ponteiro `Team` apontará para `NULL`.

Os times serão representados pela estrutura do tipo `Team` (Figura 2). Cada elemento de `Team` representa um atributo da seleção. Os atributos inteiros (*ataque*, *defesa*, *resistencia* e *velocidade*) armazenarão valores entre 0 e 100. O atributo *nome* é nominal e imutável.

2.3 Escolhendo os 16 times

Para inicialização do jogo, será fornecido um arquivo de entrada (`teams.txt`) contendo diversas seleções participantes da copa. O seu programa deverá ler este arquivo e escolher 16 times, de forma aleatória, para participar da fase de mata-mata e armazená-los em memória em uma **lista duplamente encadeada**.

Após armazenar os 16 times na lista, você poderá escolher um desses 16 times para jogar por você. Na tela de escolha de time, na qual o jogador escolherá uma seleção para jogar, o usuário poderá ver apenas **um dos quatro atributos de cada time**. A escolha do atributo que estará visível deverá ser feita de maneira aleatória e as seleções deverão estar listados por ordem em que apareceram na lista duplamente encadeada. É importante destacar que na relação de times que será apresentada ao jogador, o *nome* será omitido. A Figura 5 apresenta um exemplo de como esta relação deverá ser apresentada ao jogador (Seção 3.2).

```
1     typedef struct node {
2         Team* team;
3         struct node* left;
4         struct node* right;
5     } t_node;
6
```

Figura 1: Definição da estrutura nó que forma a árvore binária que representa o torneio.

```

1     typedef struct {
2         char* nome;
3         int ataque;      // 0 a 100
4         int defesa;      // 0 a 100
5         int resistencia; // 0 a 100
6         int velocidade;  // 0 a 100
7     } Team;
8

```

Figura 2: Definição da estrutura `Team` que representa um time (Seleção).

2.4 Arquivo de Entrada

A árvore binária será preenchida com base no arquivo `teams.txt`. Este é um arquivo de texto simples, sem formatação, que apresenta a relação de times que será utilizada na elaboração do torneio. Conforme definição, a árvore binária é formada, inicialmente, por 16 seleções distintas. Contudo, o arquivo `teams.txt` pode conter uma lista muito maior de times possíveis. Cada linha do arquivo representa um time distinto, que forma a base de times possíveis a serem utilizados no jogo. No arquivo, cada time é descrito da seguinte forma:

```

Brasil, 30, 47, 63, 70
Estados Unidos, 50, 40, 70, 100

```

Onde, os campos listados por cada linha, segundo a estrutura `Team` (Figura 2), apresentam os seguintes significados:

```

nome, ataque, defesa, resistencia, velocidade

```

2.5 Construindo a Árvore Binária

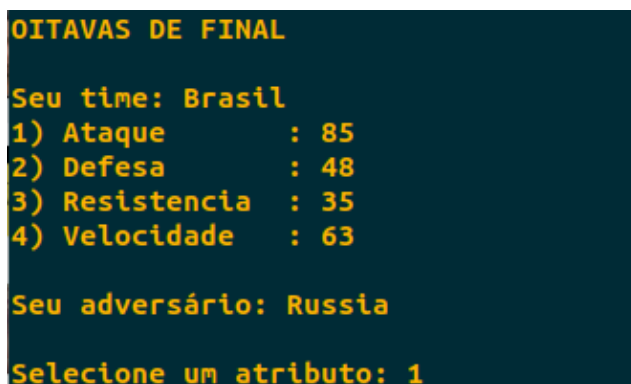
Conforme descrição apresentada na Seção 2.3, os 16 times selecionados de forma aleatória, a partir do arquivo `teams.txt`, deverão ser armazenados em memória em uma lista duplamente encadeada. Cada elemento da lista armazenará um time, representado no formato `Team` (Figura 2). Esta estrutura será utilizada para a criação da árvore binária.

Conforme descrição apresentada na Seção 2.7, a função `tree_create()` retorna um ponteiro do tipo `t_node` que aponta para o nó raiz da árvore binária, cujos os nós contém o ponteiros `team` apontando para `NULL`. Este é o estado inicial do processo de construção da árvore binária. O processo

de construção da árvore continua, envolvendo a utilização da lista duplamente encadeada (devidamente inicializada) e a árvore binária retornada pela função `tree_create()`. Neste ponto, o ponteiro `team`, da estrutura do tipo `t_node`, de cada **nó folha** da árvore binária, precisa ser apontado para seu respectivo time (`Team`) armazenado na lista duplamente encadeada.

2.6 Partida

A mecânica da partida de futebol funciona como um *super-trunfo*: você escolhe um atributo (*ataque*, *defesa*, *resistência* ou *velocidade*) para disputar com seu oponente. Vence o time que possuir o maior valor do atributo escolhido. Durante a partida, deverá ser mostrado ao jogador todas as informações do seu time. Com base nesta informação, o usuário poderá escolher qual atributo irá utilizar para enfrentar o seu oponente.



```
OITAVAS DE FINAL

Seu time: Brasil
1) Ataque      : 85
2) Defesa      : 48
3) Resistencia : 35
4) Velocidade  : 63

Seu adversário: Russia

Selecione um atributo: 1
```

Figura 3: Tela do Partida

As demais *partidas* do jogo serão controlados pelo "computador". Isto é, o seu programa sorteará de forma aleatória os atributos para que as demais Seleções disputem, definindo assim o vencedor de cada *partida*. Seguindo a definição de ordem apresentada pela árvore binária, a seleção vencedora subirá para o próximo nível do torneio e enfrentará o vencedor da "partida vizinha".

A Figura 3 apresenta um exemplo de como a tela de partida que deverá ser apresentada a cada partida realizada. Conforme pode ser observado, a informação da *etapa* em que a partida está sendo realizada precisa ser apresentada (oitavas de final, quartas de final etc). Em seguida, as informações do seu time escolhido (Seção 2.3) serão listadas, bem como, as do seu ad-

versário¹. Note que, do adversário selecionado, somente o nome será apresentado. Por fim, é apresentada a opção de escolha de qual habilidade será utilizada na partida. No exemplo apresentado pela Figura 3, a habilidade escolhida foi *ataque* (opção 1).

Observação: Como regra importante do jogo, NÃO É POSSÍVEL ESCOLHER PARA UMA *PARTIDA* O MESMO ATRIBUTO UTILIZADO NA *PARTIDA* ANTERIOR. A ideia aqui é que um mesmo atributo não pode ser utilizado por duas *partidas* consecutivas, uma vez que é necessário um *jogo* de intervalo para o seu reestabelecimento. O valor do atributo usado na *partida* anterior deverá ser representado como inutilizável na *partida* seguinte. Entretanto, o mesmo deve ser disponibilizado novamente após passada uma *partida*.

2.7 Assinatura das Funções a Serem Implementadas

Nesta seção serão relacionadas as assinaturas das funções que deverão ser implementadas neste trabalho.

- `t_node* node_create();`
 - retorna endereço para estrutura do tipo `t_node` alocada dinamicamente. Os ponteiros `team`, `left` e `right` são inicializados com o valor `NULL`;
- `Team* team_create(char* _nome, int _ataque, int _defesa, int _resistencia, int _velocidade);`
 - aloca dinamicamente memória para estrutura do tipo `Team`. Inicializa **POR CÓPIA** os atributos `nome`, `ataque`, `defesa`, `resistencia` e `velocidade`, utilizando, respectivamente, os parâmetros `_nome`, `_ataque`, `_defesa`, `_resistencia` e `_velocidade`. Ao final, a função retorna o endereço para estrutura `Team` alocada;
- `void team_free(Team* team);`
 - libera a memória alocada e referenciada por `team`;
- `t_node* tree_create();`
 - retorna o endereço para o nó raiz de uma árvore binária completa de quatro níveis. **TODOS** os nós da árvore apresentam o atributo `team` apontado para `NULL`.

¹É importante ressaltar que a ordem das partidas é definida pela árvore binária.

- `void tree_free(t_node* tree);`
 - remove de forma recursiva todos os nós presentes da árvore. A memória referente ao atributo `team` também deve ser liberada;
- `Team* match(Team* team_one, Team* team_two, int attribute);`
 - Compara o valor do atributo definido por `attribute` do `team_one` com o do `team_two`, retornando o ponteiro para o time vencedor. Em caso de empate, o ponteiro para `team_one` deverá ser retornado;
- `void tree_print_preorder(t_node* root);`
 - percorre a árvore binária em pré-ordem, imprimindo os times referenciados em `team`;

3 Funcionamento do programa

Nesta seção serão apresentadas algumas telas de funcionamento da mecânica do jogo *World Cup*.

3.1 Tela inicial

Inicialmente, o jogo deverá apresentar uma tela inicial contendo as opções definidas pela Figura 4. Escolhendo a opção **Iniciar Copa** o usuário será direcionado para a tela de **Escolha do Time** (Seção 3.2). A opção **Sair** faz o programa finalizar a sua execução.



Figura 4: Tela inicial do jogo *World Cup*.

3.2 Escolha de time

Nesta tela serão apresentados ao jogador os **16 times**, escolhidos de forma aleatória do arquivo **teams.txt** e armazenados em memória na estrutura de lista duplamente encadeada. O jogador poderá escolher um dos times relacionados. **É importante destacar que o nome dos times deverá ser omitido e apenas um atributo por time deverá ser exibido ao jogador**, conforme apresentado pela Figura 5. A escolha de qual atributo será apresentado é feita de forma aleatória. Após feita a sua escolha, o jogador irá disputar a partida!

```
Escolha seu time

Time 1:
Ataque: ?? Defesa: ?? Resistencia: ?? Velocidade: 87

Time 2:
Ataque: ?? Defesa: 16 Resistencia: ?? Velocidade: ??

Time 3:
Ataque: ?? Defesa: 36 Resistencia: ?? Velocidade: ??

Time 4:
Ataque: ?? Defesa: ?? Resistencia: 93 Velocidade: ??

Time 5:
Ataque: ?? Defesa: 22 Resistencia: ?? Velocidade: ??

Time 6:
Ataque: ?? Defesa: ?? Resistencia: 28 Velocidade: ??

Time 7:
Ataque: ?? Defesa: ?? Resistencia: 60 Velocidade: ??

Time 8:
Ataque: ?? Defesa: ?? Resistencia: ?? Velocidade: 27

Time 9:
Ataque: 27 Defesa: ?? Resistencia: ?? Velocidade: ??
```

Figura 5: Tela para escolha do time.

Observação: Por questões de limitação de espaço, a Figura 5 apresenta somente 9 times. Contudo, é mandatória a exibição de 16 times, conforme especificação.

3.3 Mecânica de Funcionamento da Copa

A Figura 6 apresenta um exemplo de como a tela de partida que deverá ser apresentada a cada partida realizada. Conforme pode ser observado, a informação da *etapa* em que a partida está sendo realizada precisa ser apresentada (oitavas de final, quartas de final, etc). Em seguida, as informações do seu time escolhido (Seção 2.3) serão listadas, bem como, as do seu adversário². Note que, do adversário selecionado, somente nome será apresentado. Por fim, é apresentada a opção de escolha de qual habilidade será utilizada no partida. No exemplo apresentado pela Figura 6, a habilidade escolhida foi *ataque* (opção 1).

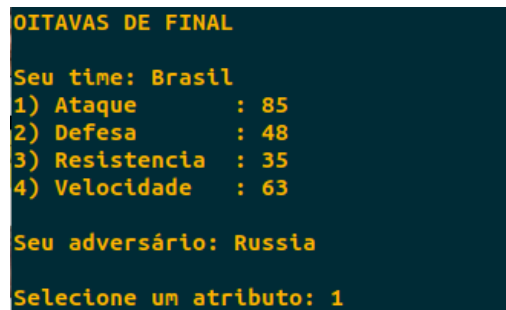


Figura 6: Tela da partida das oitavas. O atributo 1 (*ataque*) foi escolhido para o embate.

Finalizada a partida, deverá ser mostrado ao jogador o seu resultado:

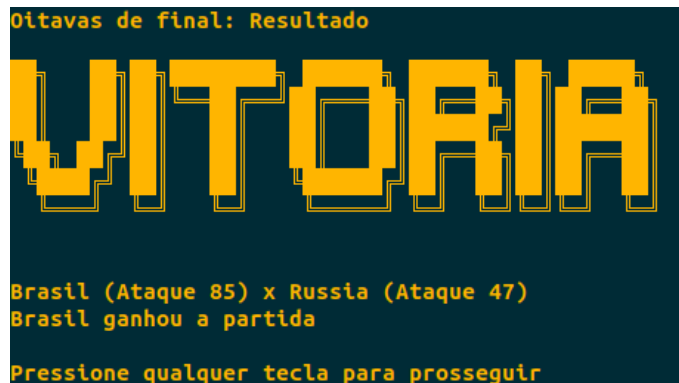


Figura 7: Tela de vitória da partida.

²É importante ressaltar que a ordem das partidas é definida pela árvore binária.

Conforme definição (Seção 2.6), na próxima *partida*, o jogador não pode selecionar o mesmo atributo utilizado na partida anterior. A Figura 8 apresenta esta restrição.

```
Quartas de Final

Seu time: Brasil
X) XX          : XX
2) Defesa      : 48
3) Resistencia : 35
4) Velocidade  : 63

Seu adversario: Itália

Selecione um atributo: 4
```

Figura 8: Tela da partida nas quartas. O atributo 1 (*ataque*) não pode ser escolhido pois foi usado na partida anterior.

```
Quartas de final: Resultado

VITORIA

Brasil (Velocidade 63) x Itália (Velocidade 55)
Brasil ganhou a partida

Pressione qualquer tecla para prosseguir
```

Figura 9: Tela de vitória, nas quartas de final.

De acordo com Figura 10, na Semifinal, o atributo que não pode ser escolhido é o 4 (*velocidade*), pois foi escolhido na partida anterior. Entretanto, o atributo 1 (*ataque*) que foi utilizado nas oitavas de final já pode ser reutilizado.

```
Semifinal

Seu time: Brasil
1) Ataque      : 85
2) Defesa      : 48
3) Resistencia : 35
XX) XX         : XX

Seu adversario: Alemanha

Selecione um atributo: 2
```

Figura 10: Tela da partida nas semifinais. O atributo 1 (*ataque*) fica disponível novamente.

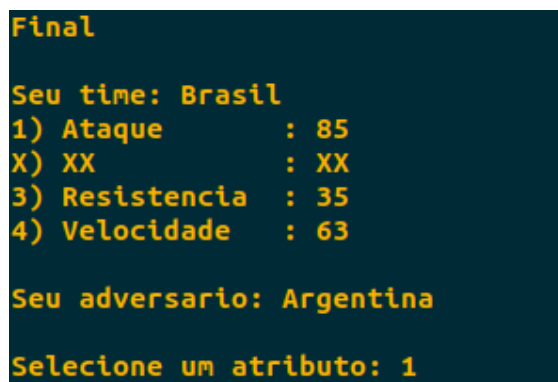
A Figura 11 apresenta a tela quando o valor do atributo escolhido pelo jogador é menor que o valor do seu oponente (derrota).

A partir da tela apresentada pela Figura 11, caso o jogador selecione a opção 1, o programa deve retornar para a tela inicial. Caso seja selecionada a opção 2, o programa deve encerrar sua execução. Além disso, devem ser mostradas todas as partidas que ocorreram **até a etapa em que o usuário perdeu**, no caso do exemplo foram mostradas todas as partidas até a *Semifinal* (etapa na qual o jogador perdeu).



Figura 11: Tela de derrota na semifinal. O jogador tem a opção de voltar ao menu principal (1) ou sair do jogo (2).

Mas é claro que não vamos deixar a Seleção Brasileira perder nesse exemplo. Digamos que ela tenha vencido o partida anterior e tenha ido para a *final*. Neste caso, A Figura 12 apresenta o resultado desta continuação.



```
Final

Seu time: Brasil
1) Ataque      : 85
X) XX          : XX
3) Resistencia : 35
4) Velocidade  : 63

Seu adversario: Argentina

Selecione um atributo: 1
```

Figura 12: Tela da partida na final. O atributo 1 (*ataque*) é escolhido para a partida.

Caso o usuário vença a final, **ele será o grande campeão mundial**. Nessa tela (Figura 13) ele tem a opção de voltar ao menu inicial (1) ou sair do jogo (2). Note que será exibido ao jogador todas as partidas que ocorreram na copa até a vitória. Perceba que existem apenas duas condições em que todas as partidas do torneio aparecem: quando o jogador ganha a copa ou quando ele perde na grande final.

```
Final: Resultado

CAMPEAO

Brasil (Ataque 85) x Argentina (Ataque 73)
Brasil ganhou a copa do mundo

Jogos da Copa
Oitavas de final:
Brasil (Ataque 85) x Russia (Ataque 47)
Itália (Defesa 64) x Croacia (Defesa 39)
Alemanha (Defesa 67) x Holanda (Defesa 52)
Colombia (Resistencia 72) x Camaroes (Resistencia 46)
Argentina (Ataque 73) x Marrocos (Ataque 23)
Uruguai (Velocidade 56) x Costa Rica (Velocidade 48)
Inglaterra (Velocidade 62) x Belgica (Velocidade 36)
França (Ataque 43) x Africa do Sul (Ataque 32)

Quartas de final:
Brasil (Velocidade 63) x Itália (Velocidade 55)
Alemanha (Defesa 67) x Colombia (Defesa 43)
Argentina (Resistencia 50) x Uruguai (Resistencia 35)
Inglaterra (Resistencia 56) x França (Resistencia 42)

Semifinal:
Brasil (Defesa 48) x Alemanha (Defesa 67)
Argentina (Ataque 73) x Inglaterra (Ataque 48)

Final:
Brasil (Ataque 85) x Argentina (Ataque 73)

[1] Voltar ao menu principal
[2] Sair
```

Figura 13: Tela de vitória da Copa.

4 Avaliação

A avaliação do projeto será composta por duas partes.

4.1 Código (70%)

- (10%) Código modularizado (makefile) e documentado;
- (20%) Implementação da estrutura árvore binária;
- (10%) Implementação da estrutura lista duplamente encadeada;
- (20%) Implementação das partidas;
- (5%) Recuperação das informações contidas no arquivo de entrada (`teams.txt`);
- (5%) Implementação de forma correta das funções definidas na Seção 2.7.

4.2 Relatório (30%)

O relatório a ser entregue deve estar OBRIGATORIAMENTE no formato PDF. Ele deve conter as seguintes informações:

- Documentação doxygen
- Descrição de como os problemas foram solucionados e implementados;
- Toda informação necessária para compilação e execução do seu programa.

5 Prazos

Data de entrega: 24/06/2018

Data de apresentações possíveis: 26/06 , 28/06, 03/07, 05/07

Deverão ser marcadas as apresentações no moodle. As vagas serão em ordem de quem pedir primeiro.

6 Observações

As seguintes observações deverão ser consideradas pelos alunos que forem fazer o trabalho:

1. O trabalho deverá ser feito (**OBRIGATORIAMENTE**) individualmente;
2. Para ser considerado entregue, o aluno fará uma apresentação do projeto implementado. No fórum da disciplina serão publicados os horários disponíveis para a apresentação dos trabalhos;
 - **Trabalhos não apresentados ganharão nota zero;**
 - **O aluno** deverá ter o conhecimento de **TODO o trabalho**. Ou seja, ter condições de responder a quaisquer questionamento referente a solução implementada e a documentação feita;
3. Utilize nomes coerentes e auto-explicativos para variáveis, funções e arquivos;
4. Todo arquivo deve conter um cabeçalho explicativo;
5. Seu código deve estar todo comentado;
6. Identação faz parte o código;
 - Será descontado 1,0 ponto do trabalho que apresentar código não indentado;
7. Serão descontados pontos dos trabalhos que apresentarem vazamento de memória (utilize o *valgrind* para verificação);
8. Os algoritmos de manipulação da estrutura de árvore devem ser **OBRIGATORIAMENTE** recursivos. O uso de algoritmo iterativo ou de operação fixa (dada a estrutura conhecida da árvore) não serão considerados;
9. O trabalho deve rodar **OBRIGATORIAMENTE** na plataforma GNU/Linux;
 - Trabalhos que não rodarem na plataforma GNU/Linux levarão nota zero;
10. Deve ser utilizado **OBRIGATORIAMENTE** a sintaxe ANSI C;

11. No relatório do trabalho deve conter as instruções para compilação do código e a relação de todas as bibliotecas utilizadas;
 - A nota do trabalho que não compila é zero. É sua responsabilidade se certificar que o código submetido está correto e compila na plataforma GNU/Linux;
12. O relatório deverá ser entregue **OBRIGATORIAMENTE** no formato de arquivo PDF;
13. Códigos copiados (entre alunos, retirados da Internet ou do repositório da disciplina) serão considerados "cola" e todos os alunos envolvidos ganharão nota zero;
14. Dúvidas sobre o trabalho deverão ser tiradas **NO FÓRUM DE DÚVIDAS DO AMBIENTE APRENDER**. Desta forma, dúvidas comuns e esclarecimentos poderão ser respondidos uma única vez para toda a turma.