6ª exercício avaliativo (EA6)
Programação II (INF16153) - UFES
11 de Julho de 2023

Nesta atividade, vamos simular um sistema baseado nos jogos da franquia Super Trunfo e na coleção de cards da elma-chips Dracomania. A ideia é simples, seu programa deve conter uma lista de cartas com seus atributos, e depois deve simular "lutas" entre essas cartas.

Cartas:
Nossas cartas serão baseadas no jogo dracomania (segue exemplo):

Todas as cartas deverão possuir as seguintes informações:

- Identificador
- Nome
- Poder de Magia
- Poder de Força
- Poder de Fogo

Usando como exemplo a primeira carta da imagem acima, temos:

- Nome: `Dragao Branco`
- Identificador: `7`
- Magia: `21`
- Força: `15`
- Fogo: `24`

Luta:
Uma luta funcionará comparando duas cartas em um de seus atributos que será selecionado. Em caso de empate a carta com menor identificador deve vencer.
Toda luta deve conter os seguintes atributos:

- Identificador da luta
- Identificadores dos lutadores 1 e 2
- Atributo a ser comparado (`M` para magia, `F` para força, `G` para fogo)

Regras de funcionamento

- Seu programa deve ter a capacidade de cadastrar uma carta (`C`), realizar uma luta (`L`) ou finalizar o programa (`F`).
- As cartas cadastradas possuem algumas restrições:
    - Todas terão nome de tamanho máximo de 50 caracteres
    - Nenhum de seus atributos pode possuir valor maior que 30
    - A soma de seus atributos não pode ser maior que 80

  Caso algum dos tópicos não seja válido a carta deve ser desconsiderada.
- Em caso de cadastro de carta com um ID já existente a carta deve ser atualizada no sistema, lembrando que a atualização requer os mesmos padrões de cadastro definidos anteriormente, ou seja caso tente cadastrar uma carta com um mesmo ID, mas fora dos padrões os valores antigos devem se manter.
- As lutas sempre se darão entre duas cartas válidas, deve se comparar o poder do atributo selecionado para a luta.
- Caso uma resulte em empate, a carta vencedora deve ser aquela com menor valor de ID

Ao finalizar o programa deve-se gerar um relatório final com:

- Número de cartas cadastradas
- Número de cartas invalidadas no cadastro
- Quantidade de lutas
- Quantidade de lutas que foram para critério de desempate (**se houver luta**)
- Nome da carta com maior número de vitórias (**se houver luta**)
- ID da Luta com maior diferença de poder (**se houver luta**)

Padrão de entrada

Todas as entradas devem ser lidas a partir da entrada padrão. Cada uma das opções devem ser lidas da seguinte forma:

Cadastro de cartas:

`C` (caractere que representa o cadastro)
`7` (identificador da carta)
`Dragao Branco` (nome da carta)
`21 15 24` (atributos, na sequência magia, força e fogo)

Exemplo de cadastro de Luta:

`L` (caractere que representa uma luta)
`1` (identificador da luta)
`7 8 M` (ID da carta 1, ID da carta 2 e o atributo de comparação, que pode assumir M, F ou G)

A execução do programa deve acabar ao ler o caractere `F`.

Padrão de saída

A saída deve ser impressa na tela de acordo com o seguinte padrão:

```
Quantidade de Cartas: 10
Quantidade de descartadas: 2
Quantidade de lutas: 5
Lutas que foram desempatadas: 0
Carta mais vitoriosa: Dragao Vermelho
Luta com Maior Diferença: 5
```

Um exemplo válido

Um exemplo completo de uma entrada e saída válidas:

Entrada	Saída

C
7
Dragao Branco
21 15 24
C
8
Dragao Vermelho
20 16 25
L
1
7 8 F
F
	`Quantidade de Cartas: 2`
	`Quantidade de descartadas: 0`
	`Quantidade de lutas: 1`
	`Lutas que foram desempatadas: 0`
	`Carta mais vitoriosa: Dragao Vermelho`
	`Luta com Maior Diferença: 1`

Regras gerais

- A atividade é **individual**. Todas as questões serão testadas e plágio não será tolerado
- Seu programa deve implementar TADs com **encapsulamento total (opaco)**
- Toda lógica de negócio deve ser implementada preferencialmente fora da sua `main()`. Em outras palavras, você só deve fazer chamadas de funções do seu TAD e/ou bibliotecas (como realizado em exercícios na sala de aula)
- Seu programa será testado com o Valgrind para detectar vazamento de memória e erros de alocação. É sua responsabilidade liberar toda memória alocada.
- **Números de ponto flutuante devem ter precisão simples e apenas duas casas decimais devem ser impressas**
- Você deve fornecer um **Makefile** que gere um arquivo executável chamado **EA6**
- **O seu programa será executado da seguinte forma:**
    - ./EA6 < entrada > saida
- Haverá **correção automática**, portanto, siga os padrões de saída corretamente
    - O corretor ignora espaços e quebras de linha
    - Porém, se você escrever informações na tela, ele retornará um erro na saída (por exemplo: “digite uma carta”). Portanto, escreva somente o que foi solicitado na tela
- Organização, modularização e boas práticas de programação são critérios fundamentais de avaliação.
- A submissão da atividade será realizada via Github de acordo com as instruções já conhecidas
    - Criar uma pasta EA6 e submeter a atividade dentro dela