Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática (ICEI)
Engenharia de Computação / Engenharia de Software
Disciplina: Algoritmos e Estruturas de Dados I

Lista de Exercícios de Registros em C

Exercício 1

Neste exercício, você deve criar um protótipo de um sistema de batalha entre guerreiros de um jogo. Para isso, implemente os itens a seguir em um módulo separado chamado `jogo`.

1.1. Defina um novo tipo de dados chamado `Guerreiro` com os seguintes campos: `ataque` (inteiro), `defesa` (inteiro), `carisma` (inteiro), `pontos_vida` (inteiro) e `id_guerreiro` (inteiro).

1.2. Escreva uma função de nome `rolaDados` que simula a rolagem de três dados de seis faces tradicionais (1 a 6) e retorna a soma dessas rolagens. Note que somar os valores resultantes da rolagem de três dados de seis faces é diferente de rolar um dado que retorna um número entre 3 e 18.

1.3. Escreva um procedimento de nome `criaGuerreiro` que recebe um `Guerreiro` por passagem de parâmetro por referência e que atribui valores aos seus campos de batalha: `ataque`, `carisma` e `defesa`, **nessa ordem**. Os seus campos de batalha devem receber um valor inteiro da função `rolaDados`. Depois, atribua um valor para o campo `pontos_vida`, que deve receber a soma dos valores retornados por três execuções da função `rolaDados`. Assuma que o campo `id_guerreiro` já foi preenchido fora da função.

1.4. Escreva uma função de nome bonusCarisma que recebe um valor de carisma como parâmetro e retorna o bônus dado por esse valor de carisma. A tabela de bônus funciona da seguinte maneira:

- `Carisma` **18**: o guerreiro é extremamente carismático e tem todo o apoio da torcida, recebendo um bônus de **+3**.
- `Carisma` **16** e **17**: o guerreiro é muito carismático e tem o apoio de quase toda a torcida, recebendo um bônus de **+2**.
- `Carisma` **14** e **15**: o guerreiro é carismático e tem o apoio de alguns torcedores, recebendo um bônus de **+1**.
- `Carisma` **6** e **7**: o guerreiro é antipático, e tem alguma torcida contra ele, recebendo uma penalidade de **-1**.
- `Carisma` **4** e **5**: o guerreiro é muito antipático, e tem quase toda a torcida contra ele, recebendo uma penalidade de **-2**.
- `Carisma` **3**: o guerreiro é extremamente antipático, e tem toda a torcida contra ele, recebendo uma penalidade de **-3**.
- Para qualquer outro valor de carisma, a sua função deve retornar **0**.

1.5. Escreva um procedimento de nome `ataca` que recebe dois `Guerreiro`s por passagem de parâmetro por referência e simula um ataque do primeiro guerreiro no segundo. O ataque é dado da seguinte maneira:

a. O primeiro guerreiro rola três dados e soma os seus valores com o seu campo `ataque` e com o seu *bônus de carisma*. Essa soma é o valor do ***golpe*** do primeiro guerreiro.
b. O segundo guerreiro rola três dados e soma os seus valores com o seu campo `defesa` e com o seu *bônus de carisma*. Essa soma é o valor do ***escudo*** do segundo guerreiro.
c. Faça ***dano*** = ***golpe*** - ***escudo***. Se o ***dano*** for maior que zero, subtraia ***dano*** dos `pontos_vida` do segundo guerreiro. Ao subtrair o ***dano***, considere que o campo `pontos_vida` não pode ter valores menores que zero.

Exercício 2

Escreva um programa que simula a batalha até a morte entre dois guerreiros. Para isso, crie dois guerreiros, um com `id_guerreiro = 1` e outro com `id_guerreiro = 2`. Depois, atribua valores aleatórios para os seus campos de batalha a partir da função `criaGuerreiro` e inicie ataques intercalados entre esses guerreiros, ou seja, comece com o guerreiro `1` atacando o `2`, depois o `2` atacando o `1`, depois o `1` atacando o `2`, e assim por diante. Para simular um ataque, use a função `ataca`. A batalha deve acabar quando um dos jogadores, o perdedor, alcançar `0` `pontos_vida`. Imprima na tela “\nVencedor: “ e logo em seguida o identificador do guerreiro vencedor (exemplo de impressão: “\nVencedor: 1”). Imprima também “\nPontos de vida restantes: ” e os pontos de vida restantes do guerreiro vencedor. Este exercício deve usar o módulo `jogo` criado no exercício anterior e deve ser implementado no arquivo `testajogo.c`.