Roteiro de Aula Prática

Rafael Augusto de Melo
18 de fevereiro de 2016

Fase 1

Objetivo: criar um protótipo de um sistema de batalha entre guerreiros de um jogo. Para isso, implemente os itens a seguir.

1. Crie 2 (dois) arquivos fonte de nomes `jogador.c` e `jogador.h` e salve-os em uma mesma pasta.

2. No arquivo `jogador.h`, defina um novo tipo de dados chamado Guerreiro com os seguintes campos: ataque (inteiro), `defesa` (inteiro), pontosVida (inteiro) e `idJogador` (inteiro).

```
// Sentinelas
#ifndef JOGADOR_H
#define JOGADOR_H
struct Guerreiro
{
    /* dados do guerreiro */
};
#endif // JOGADOR_H
```

3. No arquivo jogador.h, declare uma função de nome rolaDados, sem parâmetros e que retorne um valor inteiro. A função deve ser inserida entre os comandos de abertura e fechamento de sentinelas.

```
int rolaDados( );
```

4. No arquivo jogador.c, implemente a função rolaDados para simular 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.

5. No arquivo jogador.h, declare uma função de nome criaGuerreiro, que recebe um ponteiro para um Guerreiro como parâmetro.

   ```
   void criaGuerreiro(struct Guerreiro* g);
   ```

6. No arquivo jogador.c, implemente a função criaGuerreiro que recebe um Guerreiro e atribui valores aos seus campos de batalha. Os seus campos de batalha (ataque e defesa devem receber um valor inteiro da função rolaDados. O campo pontosVida deve receber a soma dos valores retornados por três execuções da função rolaDados.

7. No arquivo jogador.h, declare uma função de nome ataca que recebe 2 (dois) ponteiros para Guerreiros como parâmetros e deve retornar o valor do dano do ataque caso o ataque seja bem sucecido, e deve retornar 0 (zero), caso contrário.

   ```
   int ataca(struct Guerreiro* primeiro,struct Guerreiro* segundo);
   ```

8. No arquivo jogador.cc, implemente a função ataca, que simula um ataque do primeiro guerreiro no segundo. O ataque é dado da seguinte maneira:
   - O primeiro guerreiro rola três dados e soma os seus valores com o seu campo ataque. Essa soma é o valor do **golpe** do primeiro guerreiro.
   - O segundo guerreiro rola três dados e soma os seus valores com o seu campo defesa. Essa soma é o valor do **escudo** do segundo guerreiro.
   - Calcula o dano da seguinte forma: dano=golpe-escudo
   - Caso o dano for maior que zero, reduza **dano** pontosVida do segundo guerreiro e retorne o valor do dano, caso contrário, retorne 0 (zero).

Fase 2

Objetivo: Escrever um programa que simula a batalha até a morte entre dois guerreiros. Para isso, implemente os itens a seguir.

1. Crie um arquivo de nome batalha.c e salve-o junto aos arquivos criados na fase 1.

2. Crie dois guerreiros, um com idJogador 1 e outro com idJogador 2.

3. Atribua valores aleatórios para os seus campos de batalha a partir da função criaGuerreiro

4. 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.

5. A batalha deve acabar quando um dos jogadores, o perdedor, alcançar 0 ou menos pontosVida.

6. Imprima na tela o identificador do guerreiro vencedor.

Fase 3

Objetivo: Escrever um arquivo Makefile, que nos ajudará a organizar a compilação do nosso código. Em seguida, executar o nosso programa.

Um arquivo Makefile tem a seguinte estrutura básica:

```
target: dependencies
    [tab] system command
```

onde target, em geral, indica o nome de um arquivo a ser criado ou uma ação. dependencies são os arquivos gerados para a criação do arquivo target ou para a execução da ação que o target representa. system command corresponde a um ou mais comandos, precedidos de uma tabulação, que resultarão na criação do arquivo target ou na execução da ação representada por target.

1. Crie um arquivo de nome Makefile (sem extensão) e salve-o junto aos arquivos criados nas fases 1 e 2.

2. Inclua as seguintes linhas no Makefile:

```
# Todo texto apos # eh um comentario
arena : batalha.o jogador.o
    gcc batalha.o jogador.o -o arena
batalha.o : batalha.c
    gcc -c batalha.c
jogador.o : jogador.c jogador.h
    gcc -c jogador.c
clean:
    rm *.o
    rm arena
```

3. Abra o terminal, navegue até a parta onde o arquivo Makefile estiver localizado e execute o comando *make arena* para compilar os nossos arquivos.

4. Caso a compilação ocorra sem erros, execute o comando ./arena para executar o nosso jogo.

Desafio

Objetivo: Escrever um programa que simula um campeonato entre 16 guerreiros.

Este campeonato deve ser do tipo mata-mata, ou seja, eliminatório. Na primeira rodada, simule 8 batalhas entre os 16 guerreiros, em que o guerreiro 1 enfrenta o 2, o 3 enfrenta o

4, e assim por diante. Depois, simule batalhas entre os vencedores dos confrontos, ou seja, o vencedor do confronto 1 enfrenta o vencedor do confronto 2, o vencedor do confronto 3 enfrenta o vencedor do confronto 4, e assim por diante. Repita esse procedimento até chegar no campeão. Imprima o seu identificador e a sua quantidade de pontos de vida. Salve a sua implementação no arquivo campeonato.c. Adicione mais informações ao Makefile para que ele possa compilar o arquivo campeonato.c com o comando *make campeonato*.