FURG - Universidade Federal do Rio Grande
C3 - Centro de Ciências Computacionais
Algoritmos e Estruturas de Dados I
Trabalho I 4o Bim de 2022

Informações:

Trabalho Avaliado de Algoritmos do 4o bimestre de 2022
Peso: 5,0 Pontos
Grupos: <= 2
Entrega: 22/01/2023 - 23h59 - Enviar apenas código (.c)

1 - (2,0) Escreva um programa que define o tipo **Personagem** como uma *struct* de C. Este tipo contém os membros **nome** (80 caracteres), **poder** (inteiro), **defesa** (inteiro) e **vida** (inteiro). Crie dois personagens: ("Xil", 54, 12, 100) e ("Yen", 40, 20, 100). Crie também um **procedimento status()**, que recebe um personagem como parâmetro e imprime seu estado atual (todos os campos da struct). Chame este procedimento na função **main** para mostrar o *status* dos dois personagens.

Exemplo de saída:

```
Xil poder=054 defesa=012 vida=100
Yen poder=040 defesa=020 vida=100
```

2 - (2,0) Com base na atividade anterior, adicione uma função **ataque()**, que recebe como parâmetro dois personagens: p1 e **p2**. Essa função calcula o dano causado pelo ataque de **p1** sobre **p2**, usando a fórmula **dano = p1.poder - p2.defesa** (se o cálculo for negativo, o dano deve ser zero). O resultado inteiro do dano deve ser retornado. Na função **main**, calcule e mostre o dano que seria causado por um ataque de Xil a Yen, e depois o dano que seria causado pelo ataque de Yen a Xil.

Exemplo de saída:

```
Xil poder=054 defesa=012 vida=100
Yen poder=040 defesa=020 vida=100

ataque de Xil a Yen vai causar 34 de dano
ataque de Yen a Xil vai causar 28 de dano
```

3 - (2,0): Com base na atividade anterior, altere o programa principal para fazer uma batalha entre os dois personagens. Faça uma repetição enquanto os dois personagens estiverem com vida maior que zero. Dentro da repetição, calcule o dano do ataque de Xil a Yen, e depois de Yen a Xil. Então atualize a vida dos personagens subtraindo o dano recebido. Finalmente, imprima o *status* de ambos e repita. Ao final, indique quem venceu ou se ambos morreram.

Exemplo de saída:

```
Batalha:
      Xil poder=054 defesa=012 vida=072
      Yen poder=040 defesa=020 vida=066
      Xil poder=054 defesa=012 vida=044
      Yen poder=040 defesa=020 vida=032
      Xil poder=054 defesa=012 vida=016
      Yen poder=040 defesa=020 vida=-02

      Xil venceu
```

4 - (2,0): Com base na atividade anterior, adicione uma função **inimigo()**, que não recebe parâmetros. Essa função deve construir um novo personagem da raça "Zed", com poder, defesa e vida zero. Então use a função **rand()** para gerar valores aleatórios inteiros para o poder no intervalo [20, 30], para a defesa no intervalo [10, 30] e para a vida no intervalo [20, 50], devolvendo o personagem completo como valor de retorno. Faça então a batalha de Xil contra este inimigo gerado randomicamente.

Números Aleatórios:

```
#include <stdlib.h>
Semente: srand(1)
Gerador de números aleatórios:
      rand() → 1804289383
      faixa de valores inteiros gerados: 0 até RAND_MAX
Receitas de bolo:
      inteiros de 0 até 99:        rand() % 100
      inteiros de 100 até 200:     100 + rand() % (200 - 100 + 1)
```

5 - (2,0): Com base na atividade anterior, crie um grupo com cinco inimigos gerados aleatoriamente (guardados em um vetor de **Personagem**). Faça então a batalha transcorrer em *rounds*, onde o herói Xil luta contra cada inimigo ainda vivo de cada vez.