A brincadeira do "Círculo da Bomba"

Versão 1.0 – 19/09/2023

1. Introdução

A professora Ana Patrícia solicitou aos alunos da disciplina Estrutura de Dados de TSI para criarem uma versão digital de uma dinâmica que ela realiza com seus alunos do ensino médio: a brincadeira do **círculo da bomba.** Essa brincadeira é um pouco parecida com a da "dança das cadeiras", em que os participantes procuram uma cadeira para sentar quando a música para. Aquele que não encontrar cadeira, sai fora do jogo. O último participante que restar, é o campeão!

Neste problema faremos algumas adaptações em relação a dinâmica da "dança das cadeiras" para chegarmos à solução.

2. Fundamentação Teórica

A brincadeira do **círculo da bomba** começa com a definição das pessoas que irão participar (sem limite de quantidade). Os participantes vão se organizar em círculo (Figura 1). Determina-se, então, quantas pessoas serão vencedoras, com 0 < numVencedores <= nº participantes -1. Uma pessoa é escolhida aleatoriamente para iniciar a passagem da bomba de confetes. A partir do **inicializador,** uma música será tocada e a bomba será deslocada para os participantes seguintes, da direita para a esquerda, até que a música pare. Neste momento, o participante que ficou com a bomba é eliminado do jogo e a partida recomeça a partir do jogador sucessor. Enquanto a música estiver tocando, a bomba estará circulando pela roda de participantes.

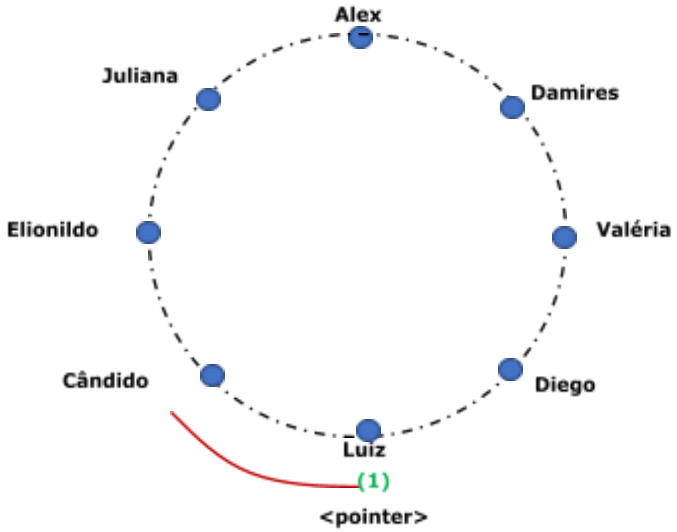
Figura 1. Ilustração da brincadeira "círculo da bomba"

Vamos considerar que a música vai ser tocada em uma quantidade de segundos **4 <= tempo <= 15** a cada iteração, e que cada segundo é o tempo da bola mudar de participante. Quando o programa identificar o(s) vencedor(es), será exibido o rastreamento dos participantes eliminados.

Nota: desconsidere a necessidade de ter que reproduzir arquivos de áudio durante a execução da simulação. Estamos interessados apenas no tempo, em segundos, que vai determinar a quantidade de avanços da bomba.

3. Simulação

Partindo do **inicializador** e movendo-se no sentido horário da lista, conta-se uma quantidade **k**[1] de avanços, previamente definida, e retira-se o participante que ficou com a bomba. O participante eliminado é anunciado e retirado do grupo. O **inicializador** passa a ser, então, o participante sucesso daquele que foi eliminado. O processo segue repetidamente até que resta apenas 1 participante.



Neste exemplo de simulação, vamos considerar como ponto de partida um **k=4** e que só teremos 1 (um) vencedor**.** De acordo com a ilustração acima, o processo inicia com a definição de **Luiz** como **inicializador**. Avançando **k** pessoas adiante, no sentido horário, o primeiro a sair é **Alex.** Este é removido da brincadeira e o **inicializador** é avançado para o elemento seguinte, **Damires.** Seguindo a lógica, para um **k=5** gerado randomicamente, o próximo elemento a ser removido é **Elionildo.** O **inicializador** é avançado para o elemento seguinte, **Juliana.** Considerando um **k=3**, o próximo elemento a ser removido é **Diego.** O processo segue adiante até que resta apenas 1 elemento. Deve ser exibido na tela o resultado do processamento do algoritmo (ver simulação a seguir).

```
Participantes: [Alex, Damires, Valéria, Diego, Luiz, Cândido, Elionildo, Juliana]
Rodada 1
Pointer: Luiz     k=4
Removido: Alex
```

[1] **k** é um número aleatório escolhido entre 1 e **n**, em que **n** é igual ao número máximo de saltos.

```
Participantes: [Damires, Valéria, Diego, Luiz, Cândido, Elionildo, Juliana]
Rodada 2
Pointer: Damires     k=5
Removido: Elionildo

Participantes: [Damires, Valéria, Diego, Luiz, Cândido, Juliana]
Rodada 3
Pointer: Juliana     k=3
Removido: Diego


Participantes: [Damires, Valéria, Luiz, Cândido, Juliana]
Rodada 4
Pointer: Luiz     k=10
Removido: Luiz

...
Vencedor após 7 rodadas: <<<< Juliana >>>>>
Percurso para a vitória:
Candido < Valeria < Damires < Luiz < Diego < Elionildo < Alex

Deseja rodar novamente o programa (S)im/(N)ão? _
```

3. Requisitos não-funcionais

Primários

- É **obrigatório** utilizar o código das estruturas de dados vistas em sala de aula para compor a solução do problema. Você pode adotar variações das estruturas de dados e/ou até mesmo adicionar novos métodos, se forem coerentes. O não cumprimento deste requisito invalida a correção do projeto.
- Os dados dos participantes também podem ser recuperados de um arquivo texto. Disponibilize um meio de carregar tais dados durante a execução do programa.
- A determinação do jogador eliminado deve seguir a lógica apresentada na Figura 1, avançando participante por participante até que a "música pare" (atingiu o número de avanços).
- O programa deve prover todas as informações e/ou leituras necessárias na tela para que o usuário possa saber o que está acontecendo, em atendimento aos requisitos estabelecidos

Secundários

- Deve haver coerência quanto à definição do nome das classes e métodos;
- Todas as situações que venham a ocasionar mal funcionamento do código devem ser pensadas e devidamente tratadas.
- Cuidado deve ser tomado para que métodos ou funções sejam definidos no local adequado, de maneira a incrementar a capacidade de reaproveitamento de código;
- O código deve contemplar os conceitos de modularização e encapsulamento
- O Programa deve permitir que, a cada início do programa, seja definido o número de pessoas que pode ser considerada **vencedora.** Na simulação, tivemos a situação de apenas 1 ganhador.

- A lógica do problema não pode ser inserida dentro do programa principal. Classes específicas devem ser pensadas para cumprir esse papel.

4. Considerações Finais

- Prepare a entrada de dados da forma mais clara e intuitiva possível para rodar o programa;
- Soluções de código "amarradas" devem ser evitadas;
- Todo tipo de validação da entrada de dados deve ser pensado pela equipe. Por exemplo, não é permitido o cadastro de duas pessoas com o mesmo nome. Outras situações serão testadas no dia da apresentação para verificar se a equipe pensou nas situações

5. Equipe

- Máximo de 3 alunos(as)