# Computação II
# Aula 1 - Revisão de Modularização

Carla Delgado - João Carlos

1. Na Copa do Mundo da Rússia, o prêmio Chuteira de Ouro é dado ao jogador que possui a melhor média de gols na competição.
   (a) Faça a função artilheiros que lê e armazena em uma matriz as seguintes informações: nome do jogador, rodada, número de gols. A leitura das informações deve ser feita jogador a jogador, e deve ser encerrada quando o nome fornecido for "fim". O item rodada informa até que rodada da Copa o jogador atuou, podendo ser um dos seguintes valores: classificatoria, oitavas, quartas, semi e final.
   (b) Faça a função mediaGols que dada a matriz de artilheiros, calcula a matriz com a média de gols dos jogadores que marcaram gols durante a Copa. Lembre que quem jogou até a fase classificatória, atuou em 3 jogos. Nas oitavas, quartas, semi e final, cada jogador joga, respectivamente, 4, 5, 6 e 7 jogos.
   (c) Faça a função principal main, onde são chamadas as funções construıdas no item anterior. Além disso, é definido o jogador que ganhará a Chuteira de Ouro. O vencedor será o jogador com maior média de gols que tenha atuado além da fase classificatória. A função deve imprimir uma mensagem informando o nome do vencedor e sua média de gols.

2. Escreva um programa que simule o sorteio de um bingo. O programa deve fazer a leitura do número total de cartelas que estão concorrendo. Para cada cartela o programa deve ler 6 números que devem estar no intervalo de 1 até 60. Os valores das cartelas devem então ser armazenadas em uma matriz.

   Quando as N cartelas forem inseridas, o programa deve sortear números até que todos os valores de uma das cartelas tenham sido sorteados (considere que uma cartela ganha o bingo quando todos os 6 números são sorteados).

   Por fim, o programa deve imprimir os números sorteados (sem repetição), a cartela vencedora e o número da cartela (a linha da cartela vencedora). Seu programa deve obrigatoriamente possuir uma função *main*, onde acontece a intereção com o usuário. Seu programa também deve possuir uma função que checa se alguma das apostas ganhou a loteria e outra função que remove elementos repetidos da lista de números sorteados.

3. **Jogo da Memória**
   O trabalho consiste em fazer o jogo da memória. O jogo será representado por uma matriz 4x4, contendo 8 pares de números (de 1 a 8), conforme exemplificado abaixo:

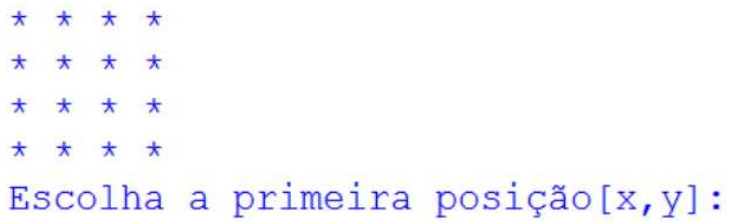
   1 4 7 8
   5 8 3 1
   5 3 2 4
   7 6 6 2

   O objetivo do jogo é descobrir a posição de todos os pares de números.
   (a) **Passo 1 - Criação da matriz**: As posições dos pares serão geradas aleatoriamente, de forma que nem você que está fazendo o programa vai saber onde estão. Para gerar posições aleatórias use a função **sample**. A função **sample** está contida no módulo **random**, o qual deve ser importado. Esta função recebe dois parâmetros: o primeiro parâmetro corresponde a uma lista, que pode conter

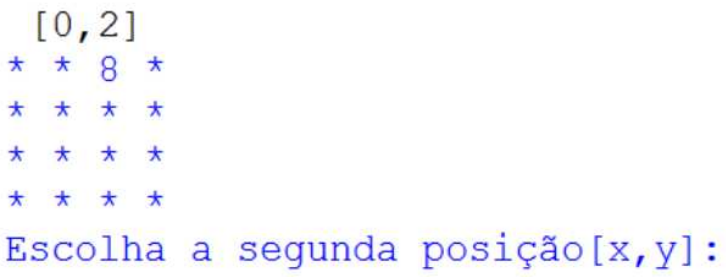
qualquer tipo de elemento. O segundo parâmetro corresponde ao número de elementos aleatórios que você quer escolher desta lista.
**Exemplo:** *random.sample([1,2,3,4,5],1)*

**Resposta:** [4]
Neste exemplo será retornado 1 (segundo parâmetro) número aleatório dentre os da lista [1,2,3,4,5] (primeiro parâmetro). Utilizando esta função você consegue escolher posições aleatórias para colocar os pares de 1 a 8.

(b) **Passo 2 - Interação com o usuário**: O jogo começa mostrando a seguinte tela ao usuário:

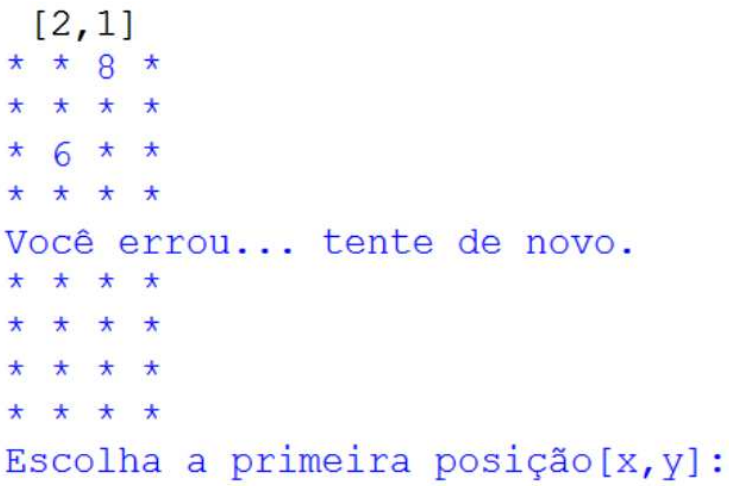



Nesta tela a matriz $4 \times 4$ é mostrada com as posições cobertas por "*". O programa deve pedir ao usuário para escolher a primeira posição a abrir. A posição digitada deve ter o formato [x,y]. Suponha que o usuário escolheu a posição [0,2], conforme mostrado abaixo:




Neste exemplo a posição [0,2] foi descoberta, mostrando o número que está nela, que no caso é o número 8. Após isso, o programa deve pedir ao usuário que escolha a segunda posição a abrir.

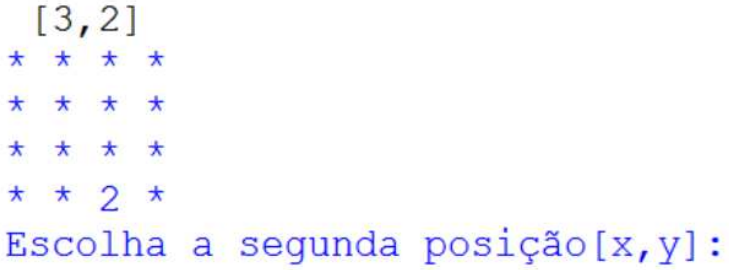
Suponha que o usuário escolheu a posição [2,1]. Conforme mostrado a seguir:




Neste exemplo a posição [2,1] foi descoberta, mostrando o número 6. Como 8 e 6 não formam um par de números iguais, o programa deve informar ao usuário que ele errou, pedindo para ele tentar de novo.

Toda vez que o usuário errar, as posições que haviam sido abertas na tentativa devem ser fechadas novamente. Observe no exemplo acima que a matriz voltou a ficar somente com "*".

Após cada tentativa de se descobrir um par, o programa deve novamente pedir ao usuário para digitar a primeira e segunda posição da próxima tentativa, começando-se pela primeira, conforme mostrado na figura acima. Suponha que nessa segunda jogada o usuário digite [3,2], conforme mostrado abaixo:




Suponha agora que o usuário escolha [1,3] como segunda posição, conforme mostrado a seguir:

```
 [1,3]
* * * *
* * * 2
* * * *
* * 2 *
Parabéns! Você acertou
Escolha a primeira posição[x,y]:
```

Observe que dessa vez o usuário acertou o par, ou seja, descobriu as duas posições do número 2. Neste caso o programa deve mostrar uma mensagem parabenizando o usuário pelo acerto e pedir a primeira posição da próxima jogada.

Quando o usuário acerta um par, os números não são mais cobertos por "*", ou seja, o par continua descoberto, mostrando que estas posições já foram abertas.

O programa deve continuar rodando até que o usuário acerte todos os pares, ficando a matriz totalmente aberta, conforme mostrado abaixo:

```
7 3 8 1
5 7 1 2
4 6 5 8
3 6 2 4
Parabéns! Você acertou
Parabéns!! Você conseguiu descobrir todas as casas em 19 jogadas!
```

Ao final do jogo, o programa deve informar que o usuário conseguiu descobrir todos os pares e deve informar em quantas jogadas ele o fez, onde cada jogada corresponde ao chute de duas posições.

(c) **Validações a serem feitas:** Duas validações devem ser feitas no seu programa, conforme discriminadas abaixo:

- **Posição fora do limite da matriz:** A matriz é $4 \times 4$, portanto as linhas e as colunas vão de 0 a 3. Caso o usuário digite uma posição que não está no limite da matriz, ou seja, algum número de linha ou coluna que não esteja entre 0 e 3, inclusive, o programa deve mostrar uma mensagem de "posição inválida" e pedir ao usuário para digitar a posição novamente, conforme mostrado no exemplo abaixo:

```
* * 8 *
* * * 2
* 6 * 8
* 6 2 *
Escolha a primeira posição[x,y]:
 [0,6]
Posição inválida. Escolha a primeira posição[x,y]:
```

Neste caso, num ponto qualquer do jogo, o usuário tentou acessar a posição [0,6], porém não existe a coluna 6 na matriz, e portanto esta é uma posição inválida.
Observe que se a posição inválida foi digitada no momento em que se pedia a primeira posição, é a primeira posição que deve ser pedida novamente. Caso a posição inválida seja digitada

quando era para escolher a segunda posição, então será a segunda posição que deve ser digitada novamente.

- **Posição já aberta**: Caso o usuário digite uma posição que já está aberta, o programa também deve considerar esta posição como inválida, pedido que uma nova posição seja digitada. Esta situação está exemplificada abaixo:

```
* * 8 *
* * * 2
* 6 * 8
* 6 2 *
Escolha a primeira posição[x,y]:
 [0,2]
Posição inválida. Escolha a primeira posição[x,y]:
```

Observe que no exemplo acima o usuário escolheu a posição [0,2], porém esta já está descoberta e contém o número 8.
Outro exemplo de posição inválida é o seguinte:

```
Posição inválida. Escolha a primeira posição[x,y]:
 [0,0]
7 * 8 *
* * * 2
* 6 * 8
* 6 2 *
Escolha a segunda posição[x,y]:
 [0,0]
Posição inválida ou já escolhida. Escolha a segunda posição[x,y]:
```

Neste caso, o usuário escolheu primeiramente a posição [0,0] que ainda não havia sido aberta, obtendo o número 7. Porém, na hora de escolher a segunda posição, ele escolheu novamente a posição [0,0], o que não pode acontecer.