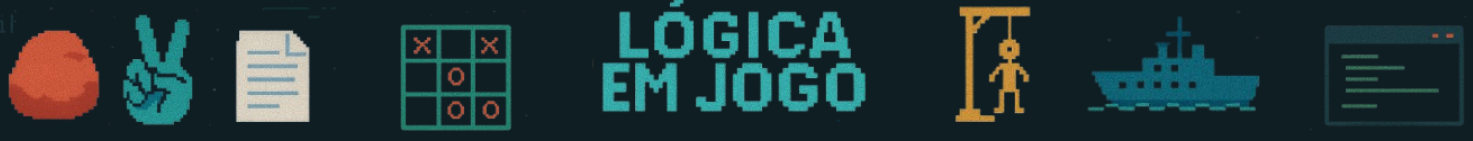


Sessão 03

Pedra, papel, tesoura, lagarto, spock

Professor Thiago Goveia

Objetivos da sessão:

- Revisar os recursos básicos da linguagem C: entrada e saída, estruturas condicionais, laços, vetores e matrizes
- Utilizar números aleatórios para introduzir a ideia de "sorte" no jogo.
- Implementar diferentes versões do jogo levando em consideração diferentes estratégias e recursos, considerando as vantagens e desvantagens de cada uma.

Materiais:

- Repositório do projeto criado na Sessão 01 no GitHub.

Sumário:

Conteúdo da sessão:

INSTITUTO FEDERAL
Minas Gerais

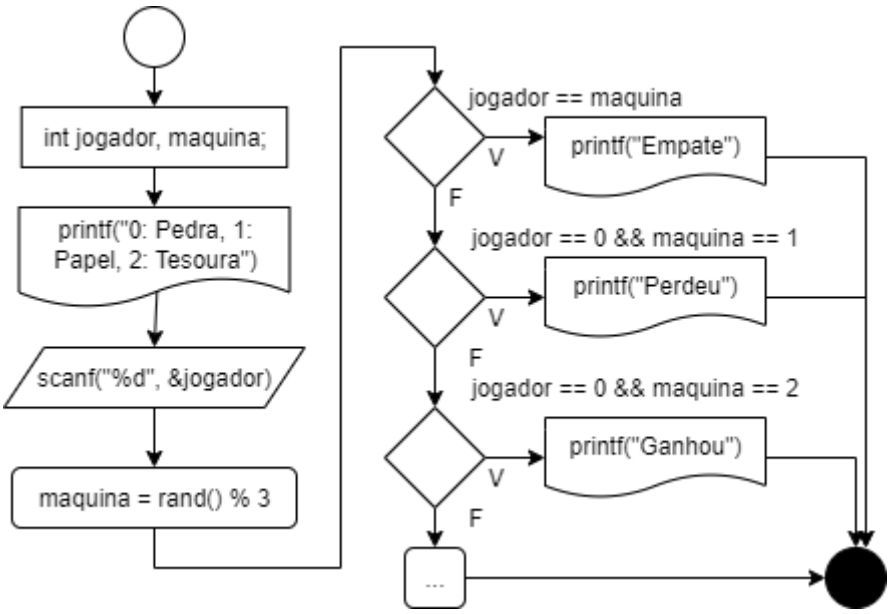
1. Introdução

1.1 As Regras do jogo clássico

Antes de programar, precisamos ter certeza de que entendemos a lógica. O jogo "Pedra, Papel e Tesoura" possui três regras de vitória:

- Pedra ganha de Tesoura (amassando-a).
- Tesoura ganha de Papel (cortando-o).
- Papel ganha de Pedra (embrulhando-a).
- Se ambos os jogadores escolherem a mesma opção, ocorre um **empate**.

A seguir está o esboço do fluxograma da sessão 02, o qual não considera a repetição.



1.2 Atualização no controle de versão

Na sessão de hoje, toda a modificação que fizermos no código será salva no github. Isso nos ajudará a desenvolver de forma colaborativa e manterá o backup de cada versão.

2. A Caixa de Ferramentas em C para o Jogo

Para construir nosso jogo, usaremos cinco categorias de ferramentas essenciais da linguagem C :

2.1 Entrada e Saída (printf e scanf)

- printf(): Exibe mensagens na tela para o usuário (ex: "Escolha sua jogada:").
- scanf(): Lê um valor digitado pelo usuário e o armazena em uma variável.

2.2 Geração de Números Aleatórios (rand e srand)

- rand(): Gera um número inteiro pseudo-aleatório. É incluída pela biblioteca **stdlib.h**.
- Para limitar o resultado (ex: entre 0 e 2), usamos o operador de módulo (%): rand() % 3.
- srand(time(NULL)): "Alimenta" a função rand() com uma semente baseada no tempo atual, garantindo que os números sejam diferentes a cada execução do programa. Deve ser usada apenas uma vez, no início do código. A função time é incluída pela biblioteca **time.h**.

2.3 Estruturas Condicionais (if/else if/else)

É o cérebro do nosso jogo. Usamos essa estrutura para comparar a jogada do jogador com a do computador e aplicar as regras para determinar o vencedor.

2.4 Laços de Repetição (while ou do-while)

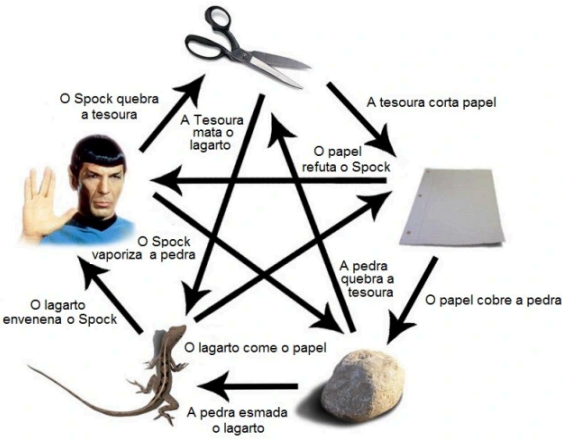
Para que o jogo não termine após uma única rodada, colocamos toda a lógica dentro de um laço que continua executando enquanto o jogador desejar continuar.

2.5 Vetor e Matriz ([] e [][])

Utilizamos estas estruturas de dados para separar a **lógica do jogo** dos **dados do jogo**. Ao utilizar vetor e matriz, aprimoramos o código, deixando-o mais limpo, mais fácil de manter e escalável/genérico.

3. Mão na Massa! Implementando Pedra, Papel, Tesoura, Lagarto Spock

Pedra-papel-tesoura-lagarto-Spock é uma expansão do jogo clássico, incluindo outras duas armas adicionais: o lagarto (formado pela mão igual a uma boca de fantoche) e Spock (formada pela saudação dos vulcanos em Star Trek). As regras são as seguintes:



- Tesoura corta papel
- Papel cobre pedra
- Pedra esmaga lagarto
- Lagarto envenena Spock
- Spock esmaga tesoura
- Tesoura decapita lagarto
- Lagarto come papel
- Papel refuta Spock
- Spock vaporiza pedra
- Pedra amassa tesoura

3.1 Versão 04 - Utilizando apenas ifs (aninhados ou não)

Vamos adicionar "Lagarto" e "Spock" ao jogo, seguindo as regras listadas acima.

Instruções:

1. Defina um número para cada jogada: 1-Pedra, 2-Papel, 3-Tesoura, 4-Lagarto, 5-Spock.
2. Adapte o código da Versão 02 ou Versão 03 para contemplar as novas regras.
3. Avalie as conclusões da equipe juntamente com a turma.

INSTITUTO FEDERAL
Minas Gerais

3.2 Versão 05 - Utilizando vetores

Vamos voltar ao jogo clássico (3 opções) e aprender uma técnica mais elegante. Em vez de escrever as regras no código, vamos armazená-las em um vetor (array). Após implementar esta versão para o jogo clássico, implemente também para o jogo com Lagarto e Spock.

Instruções:

1. Vamos usar 1-Pedra, 2-Papel, 3-Tesoura.
2. A ideia é criar um vetor onde o *índice* representa uma jogada e o *valor* armazenado nesse índice é a jogada que a derrota.
3. Declare o vetor no início do seu código. Como Pedra (1) perde para Papel (2), Papel (2) perde para Tesoura (3), e Tesoura (3) perde para Pedra (1), o vetor fica assim: `int perdePara[1] = {0, 2, 3, 1};` (A posição 0 não é usada para manter a relação direta entre o número da jogada e o índice).
4. Substitua a lógica `if/else` complexa pela verificação do vetor.
5. Como poderíamos reduzir o tamanho do vetor `perdePara`?

3.3 Versão 06 - Utilizando matrizes

Para criarmos soluções mais abrangentes e elegantes, a solução mais poderosa é uma matriz de decisão, que funciona como uma "tabela de consulta" de resultados. Após implementar esta versão para o jogo clássico, implemente também para o jogo com Lagarto e Spock.

Instruções:

1. Crie uma matriz 3x3. As linhas serão a jogada do jogador e as colunas a do computador. O valor no cruzamento `[linha][coluna]` indicará o resultado. Usaremos `1` para vitória do jogador, `-1` para derrota e `0` para empate.
2. Declare e inicialize a matriz de regras no início do seu código.
3. **Implementação:** Dentro do laço do jogo, a lógica de decisão se resume a uma consulta na matriz.

4. Indo além! (Para casa)

4.1 Meu jogo, minhas regras

Pense em uma nova regra que poderia ser incluída no jogo clássico ou Lagarto-Spock.

Próximas sessões: Pedra, Papel, Tesoura com Arduino