

*Davi Justino
Samir Guimarães*

JOCO NIM PROGRAMA

Origens Do Nim

O Jogo NIM tem origens antigas e é encontrado
em diversas culturas. A versão moderna foi
formalizada por Charles Bouton em 1901. A
matemática por trás do jogo oferece insights
fascinantes sobre teoria dos jogos e estratégias
vencedoras.

Regras

O jogo Nim é um jogo de estratégia simples, geralmente jogado com um conjunto de objetos (como palitos, pedras, ou fichas) distribuídos entre dois jogadores. As regras básicas são as seguintes.

1. Configuração Inicial: Comece com um número específico de objetos (como palitos) em uma pilha ou várias pilhas.
2. Turnos dos Jogadores: Os jogadores alternam turnos, removendo um número qualquer de objetos de uma única pilha em cada jogada.
3. Objetivo: O jogador que remove o último objeto ganha o jogo.

Regras Detalhadas

Jogo NIM



Pilha 1

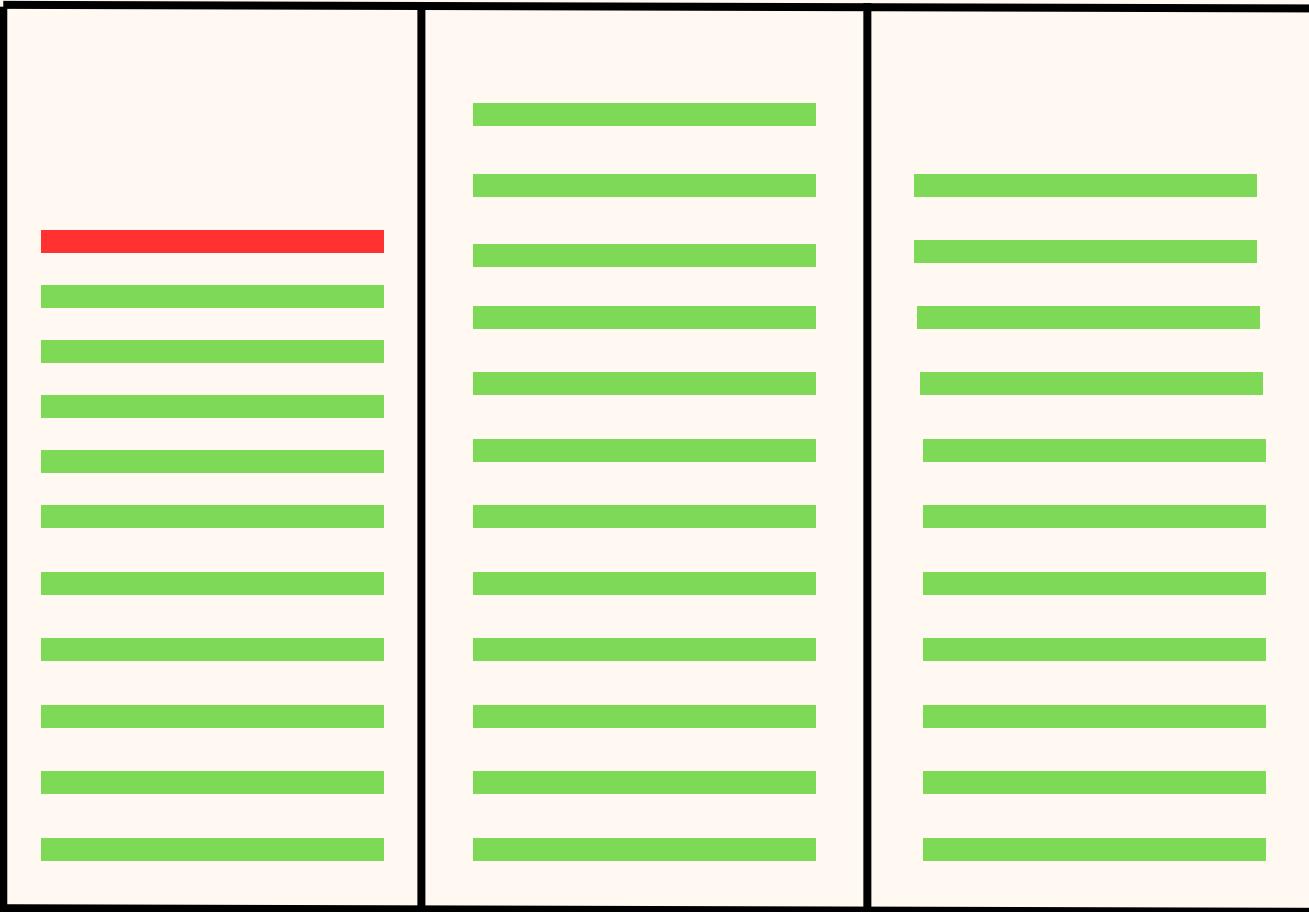
Pilha 2

Pilha 3

Jogador x Computador

O jogo começa com um número inicial de peças dispostas em pilhas. Dois jogadores alternam na retirada de peças, cada um podendo retirar uma ou mais peças de uma única pilha ou até mesmo a pilha inteira. O jogador que pegar a última peça vence o jogo.

Jogo NIM



Pilha 1

Pilha 2

Pilha 3

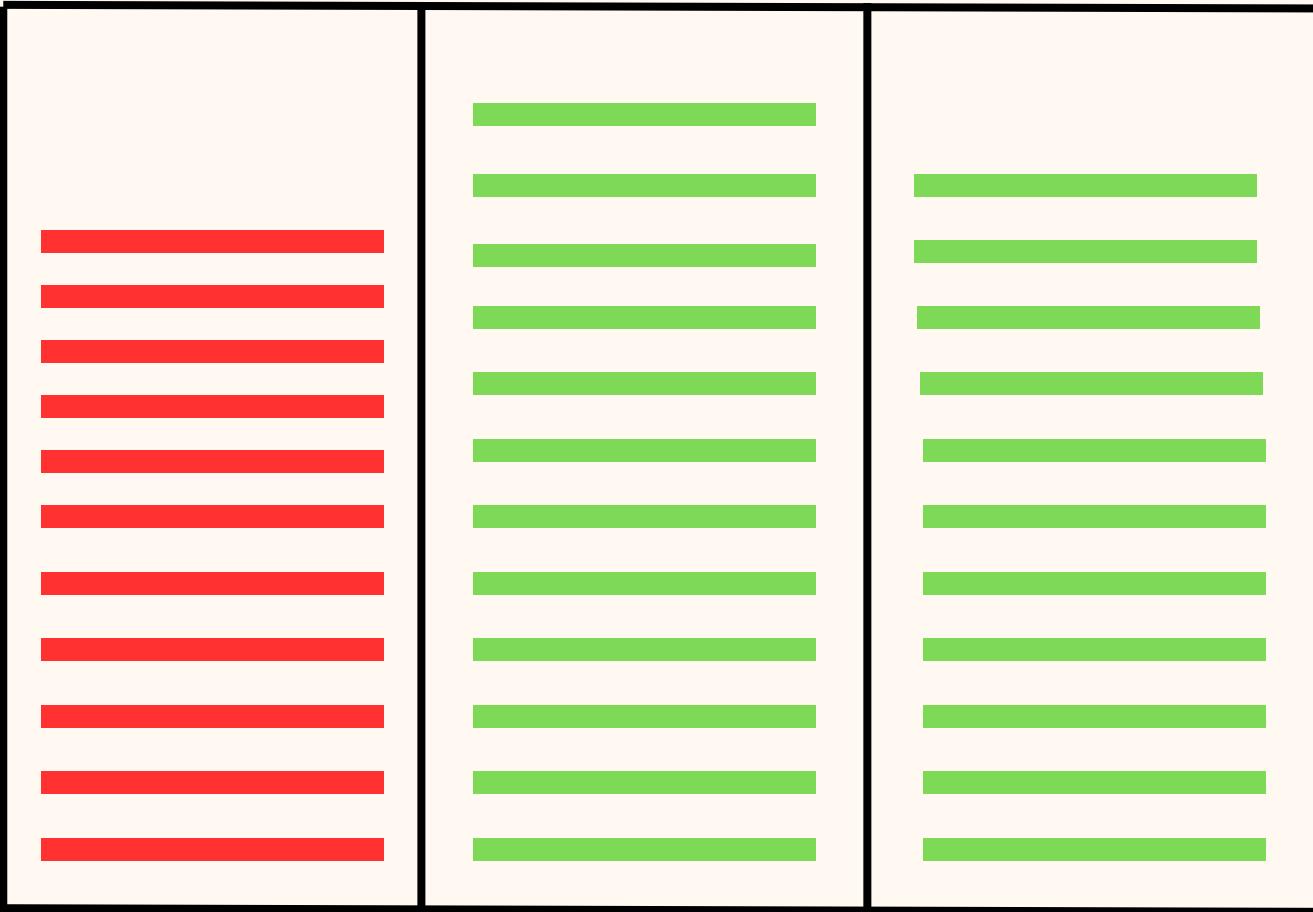
Jogador x Computador

Partida 1

Jogador

O jogo começa com um número inicial de peças dispostas em pilhas. Dois jogadores alternam na retirada de peças, cada um podendo retirar uma ou mais peças de uma única pilha ou até mesmo a pilha inteira. O jogador que pegar a última peça vence o jogo.

Jogo NIM



Pilha 1

Pilha 2

Pilha 3

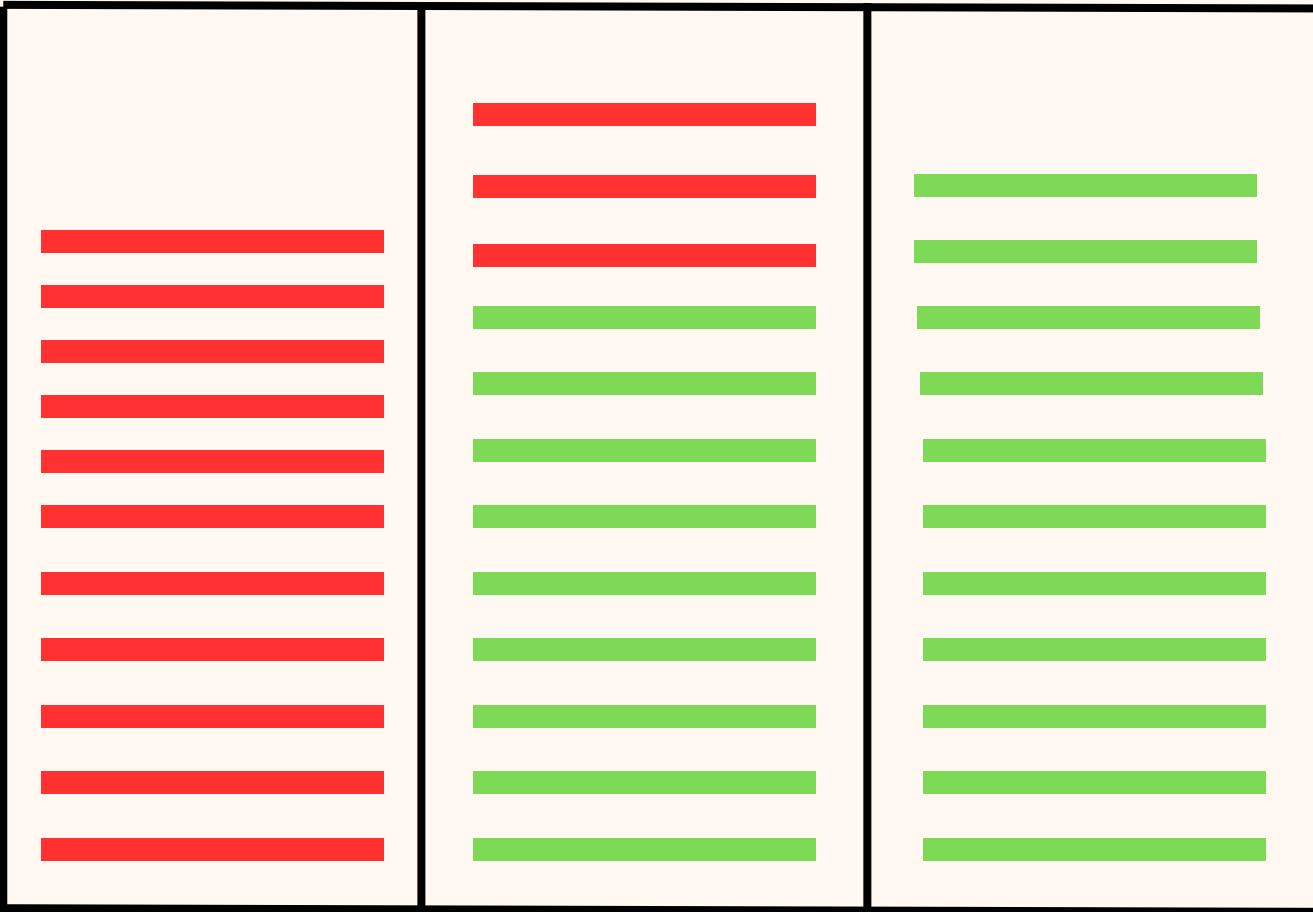
Jogador x Computador

Partida 2

Computador

O jogo começa com um número inicial de peças dispostas em pilhas. Dois jogadores alternam na retirada de peças, cada um podendo retirar uma ou mais peças de uma única pilha ou até mesmo a pilha inteira. O jogador que pegar a última peça vence o jogo.

Jogo NIM



Pilha 1

Pilha 2

Pilha 3

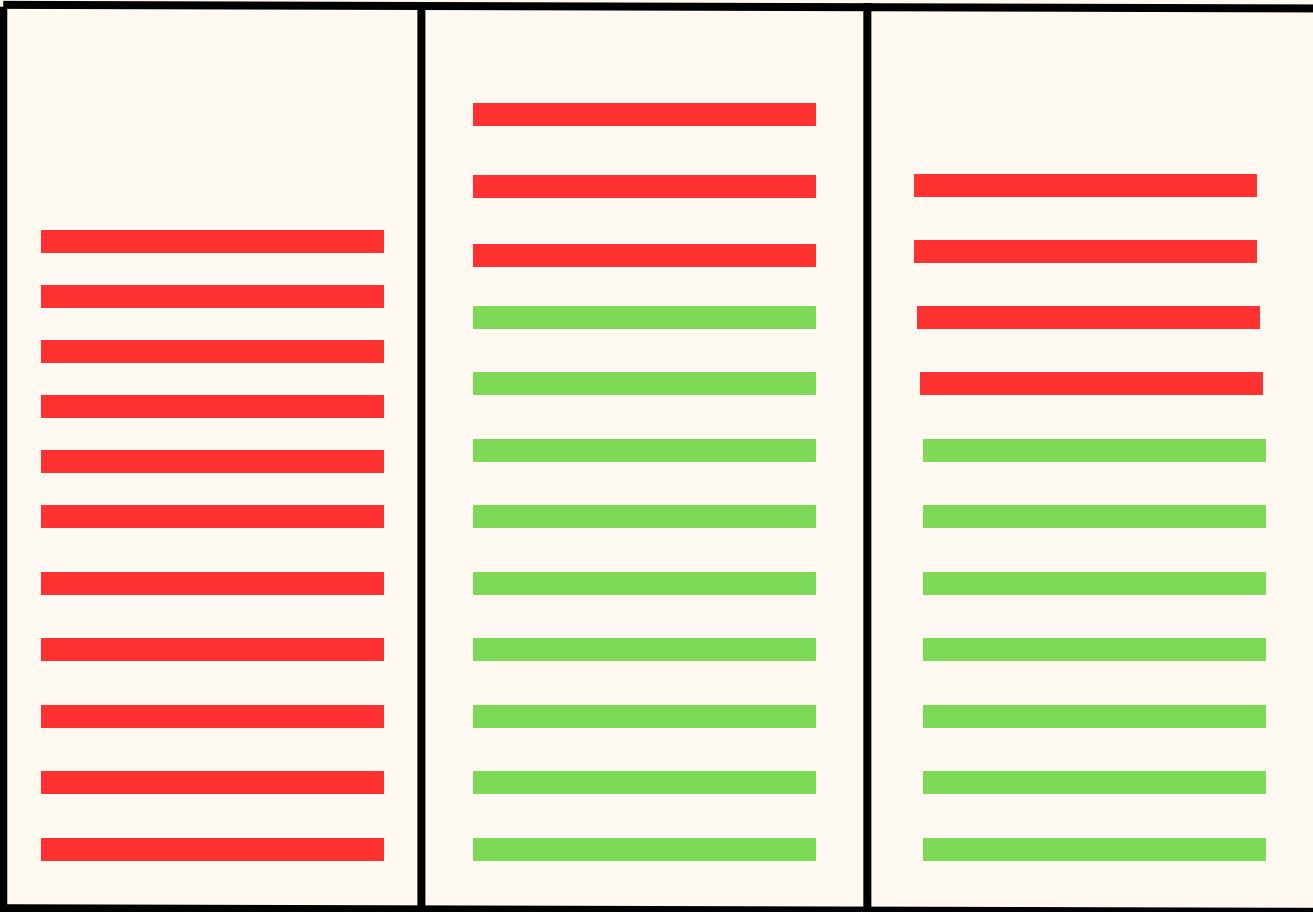
Jogador x Computador

Partida 3

Jogador

O jogo começa com um número inicial de peças dispostas em pilhas. Dois jogadores alternam na retirada de peças, cada um podendo retirar uma ou mais peças de uma única pilha ou até mesmo a pilha inteira. O jogador que pegar a última peça vence o jogo.

Jogo NIM



Pilha 1

Pilha 2

Pilha 3

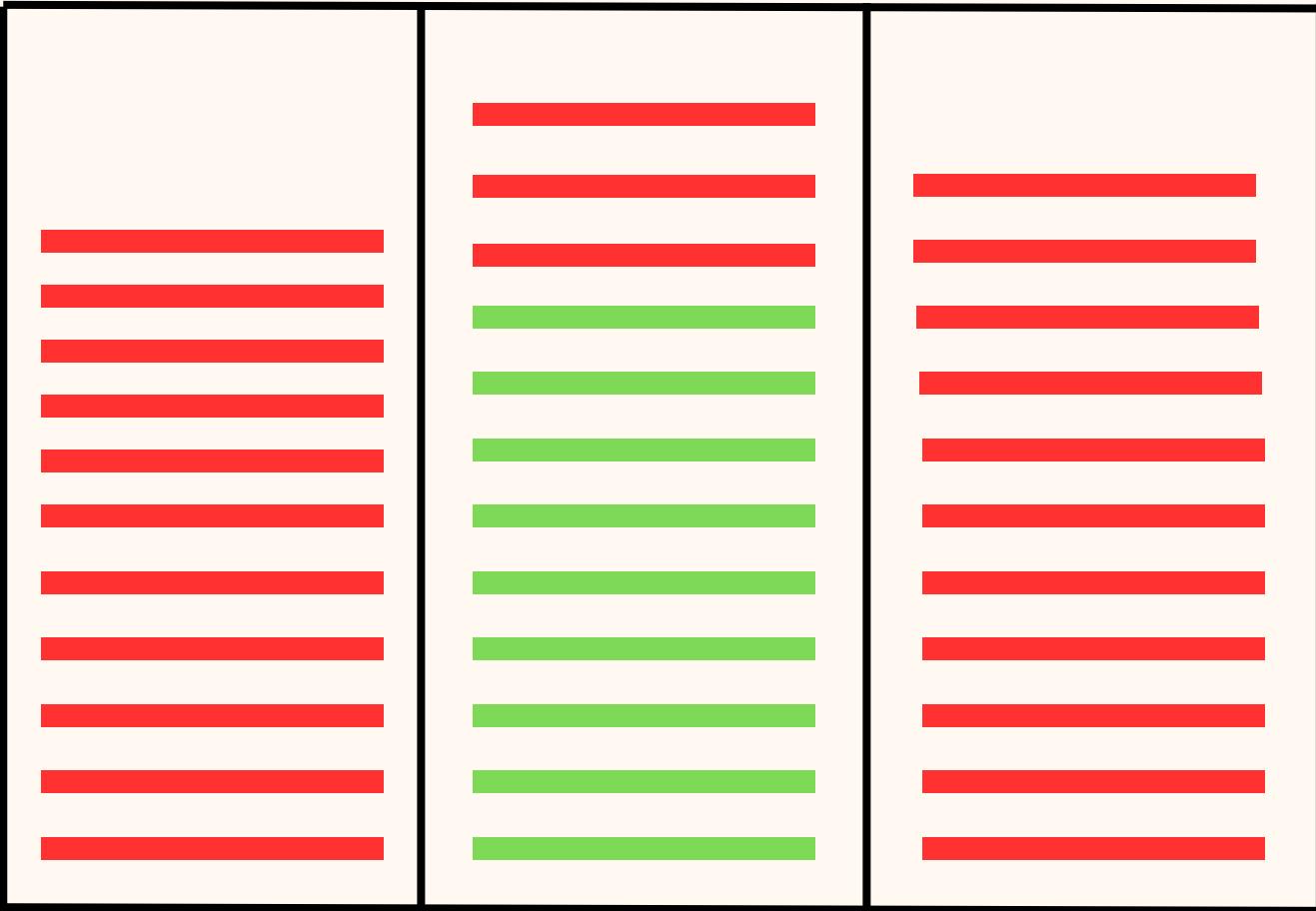
Jogador x Computador

Partida 4

Computador

O jogo começa com um número inicial de peças dispostas em pilhas. Dois jogadores alternam na retirada de peças, cada um podendo retirar uma ou mais peças de uma única pilha ou até mesmo a pilha inteira. O jogador que pegar a última peça vence o jogo.

Jogo NIM



Pilha 1

Pilha 2

Pilha 3

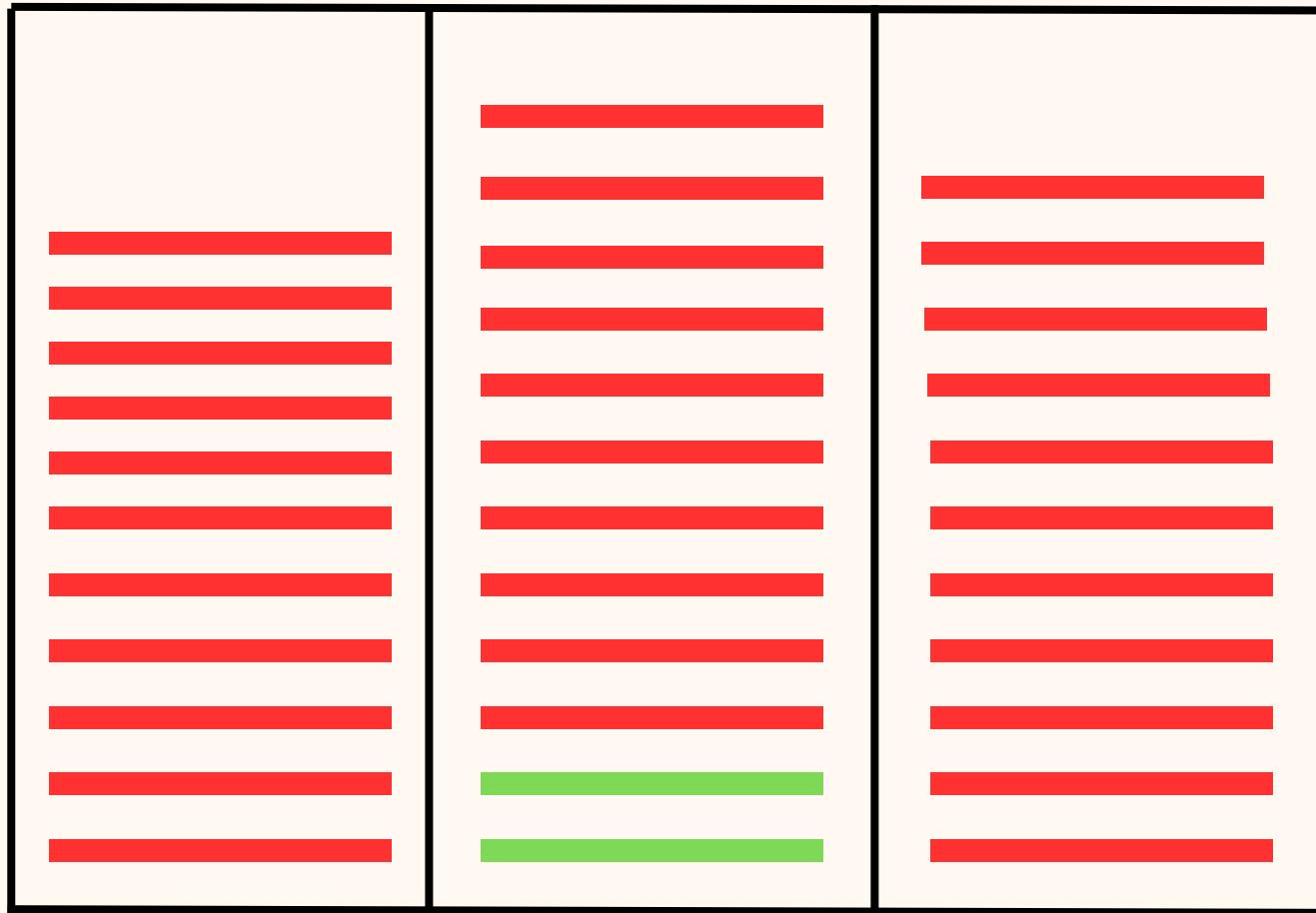
Jogador x Computador

Partida 5

Jogador

O jogo começa com um número inicial de peças dispostas em pilhas. Dois jogadores alternam na retirada de peças, cada um podendo retirar uma ou mais peças de uma única pilha ou até mesmo a pilha inteira. O jogador que pegar a última peça vence o jogo.

Jogo NIM



Pilha 1

Pilha 2

Pilha 3

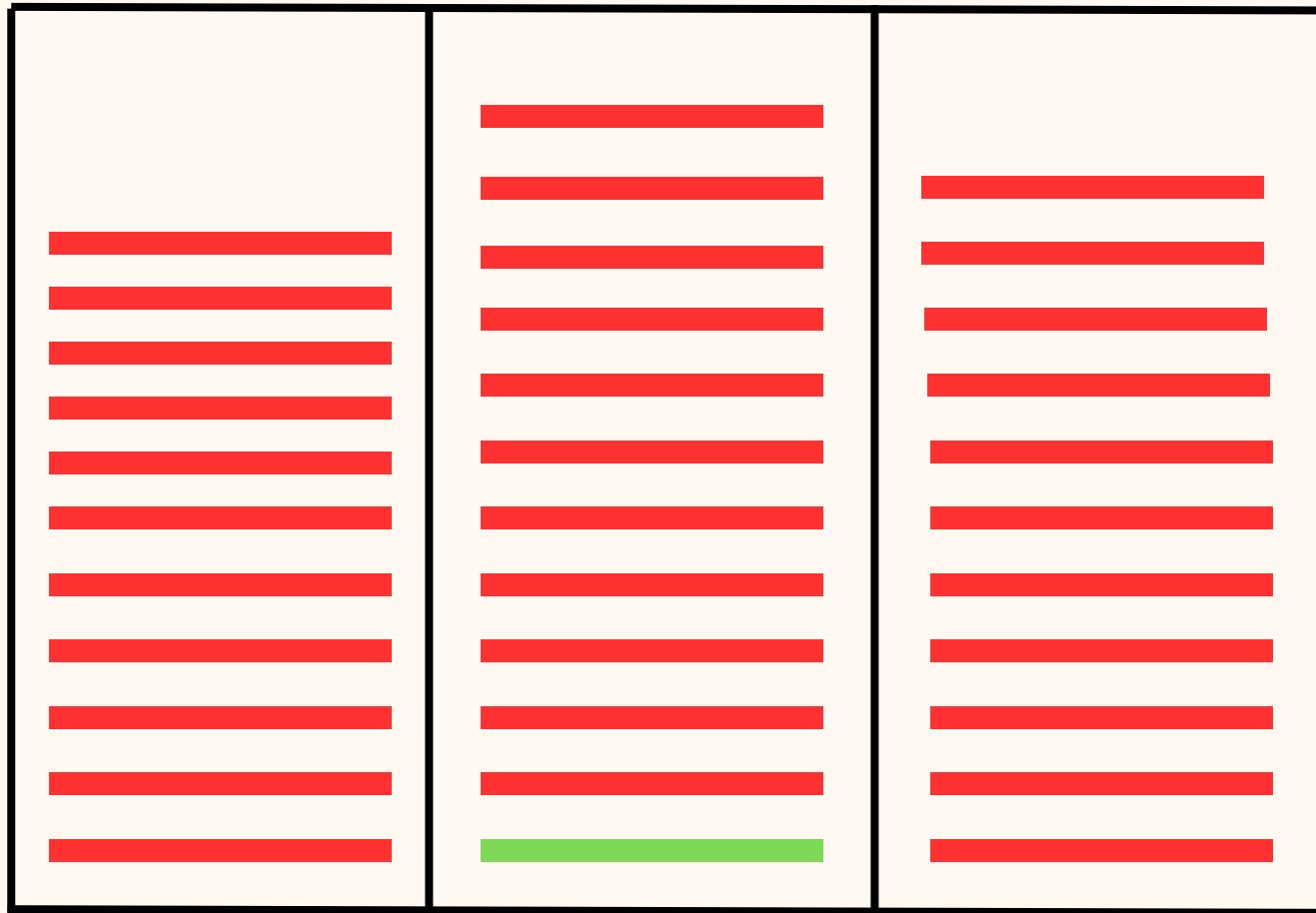
Jogador x Computador

Partida 6

Computador

O jogo começa com um número inicial de peças dispostas em pilhas. Dois jogadores alternam na retirada de peças, cada um podendo retirar uma ou mais peças de uma única pilha ou até mesmo a pilha inteira. O jogador que pegar a última peça vence o jogo.

Jogo NIM



Pilha 1

Pilha 2

Pilha 3

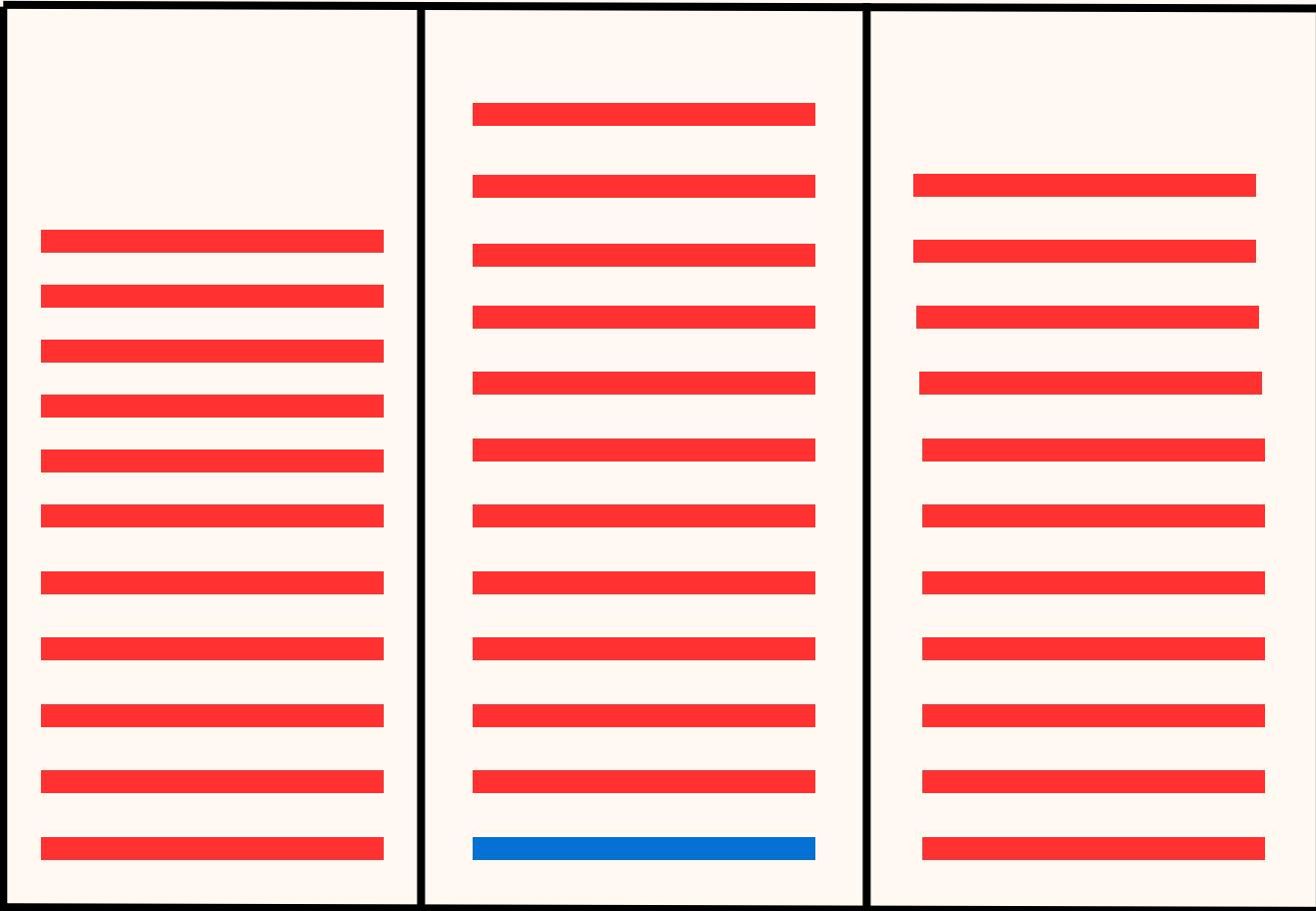
Jogador x Computador

Partida 7

Jogador

O jogo começa com um número inicial de peças dispostas em pilhas. Dois jogadores alternam na retirada de peças, cada um podendo retirar uma ou mais peças de uma única pilha ou até mesmo a pilha inteira. O jogador que pegar a última peça vence o jogo.

Jogo NIM



Pilha 1

Pilha 2

Pilha 3

Jogador x Computador

Partida 8

Computador

O jogo começa com um número inicial de peças dispostas em pilhas. Dois jogadores alternam na retirada de peças, cada um podendo retirar uma ou mais peças de uma única pilha ou até mesmo a pilha inteira. O jogador que pegar a última peça vence o jogo.

Análise Matemática: **MMC MDC**

O Jogo NIM, também conhecido com jogo das Pegadas, é um jogo milenar com regras simples, mas uma matemática intrigante por trás de cada jogada. Nesse slide, vamos desvendar os segredos matemáticos do NIM, explorando os conceitos de Máximo Divisor Comum (MDC) e Mínimo Múltiplo Comum (MMC) e como eles influenciam as estratégias vencedoras.

A matemática por trás do NIM se baseia em conceitos básicos de aritmética, como divisibilidade, múltiplos e fatores. Ao compreender esse conceitos, os jogadores podem desenvolver estratégias para identificar posições vencedoras e perdidas, aumentando suas chances

Posições Perdidas (P):

**Representações por múltiplos de 4
($4n$, onde n é um número natural).**

Nestas posições, o jogador que joga em seguida sempre terá uma jogada vencedora, retirando peças para deixar a pilha com um oponente.

Posições Vencedoras (G):

Representadas por todos os números que não são múltiplos de 4 (1, 2, 3, 5, 6, 7, 9, 10, 11, etc.). Nestas posições, o jogador que joga em seguida precisa retirar peças para deixar a pilha com um número que não seja múltiplo de 4 para o oponente, garantindo assim a vitória na próxima jogada.

Por dentro do jogo

O jogo Nim é interessante porque tem uma estratégia vencedora bem definida quando jogado corretamente:

Teoria dos Números em Jogo Nim: A chave para ganhar o Nim está na manipulação das pilhas de modo a deixar o oponente em uma posição onde ele não possa evitar perder no seu próximo turno.

Método de Binary Nim: Em uma versão simplificada, um jogador pode aplicar uma estratégia chamada "Nim binary" para determinar a melhor jogada com base nas operações binárias no número de objetos em cada pilha.

3: 011
4: 100
5: 101

Representação Binária: Cada pilha de objetos é representada por seu número de objetos em forma binária.

Exemplo: Se as pilhas têm 3, 4 e 5 objetos, suas representações binárias são:.

XOR (Exclusive OR): Calcular a soma XOR (ou exclusiva) de todos os números binários das pilhas. Isso é feito comparando cada dígito binário correspondentes dos números, onde $1 \text{ XOR } 1 = 0$, $0 \text{ XOR } 0 = 0$ e $1 \text{ XOR } 0 = 1$ (sem transporte). Exemplo:

$$3 \text{ (011)} \text{ XOR } 4 \text{ (100)} = 7 \text{ (111)} \\ 7 \text{ (111)} \text{ XOR } 5 \text{ (101)} = 2 \text{ (010)}$$

Se o resultado do XOR for 0, o jogador que acabou de jogar está em uma posição vantajosa, pois está em uma "posição segura". Se não for 0, o jogador atual deve tentar transformar a situação em uma "posição segura"

Basicamente, precisamos desenvolver
estratégias para que o último palito
seja o seu, normalmente o primeiro
jogador acaba sendo o ganhado, desde
que ele saiba como manipular,
gerenciar os seus palitos

Importância Do Jogo



Segundo Rego & Rego (1999), os jogos didáticos ajudam as crianças a desenvolverem a habilidade de trabalhar com os outros para alcançar um fim comum, de tomar decisões baseadas em planos de ação e de negociar suas concepções com a de seus colegas. Também ajudam a desenvolverem a coordenação motora e a auto-estima.

“Para crianças pequenas, os jogos são as ações que elas repetem sistematicamente, mas que possuem um sentido funcional (jogos de exercícios), isto é, são fontes de significados e, portanto, possibilitam compreensão, geram satisfação, formam hábitos que se estruturam num sistema. Essa repetição funcional também deve estar presente na atividade escolar, pois é importante no sentido de ajudar a criança a perceber regularidades” PCN’s (1997, p. 48).

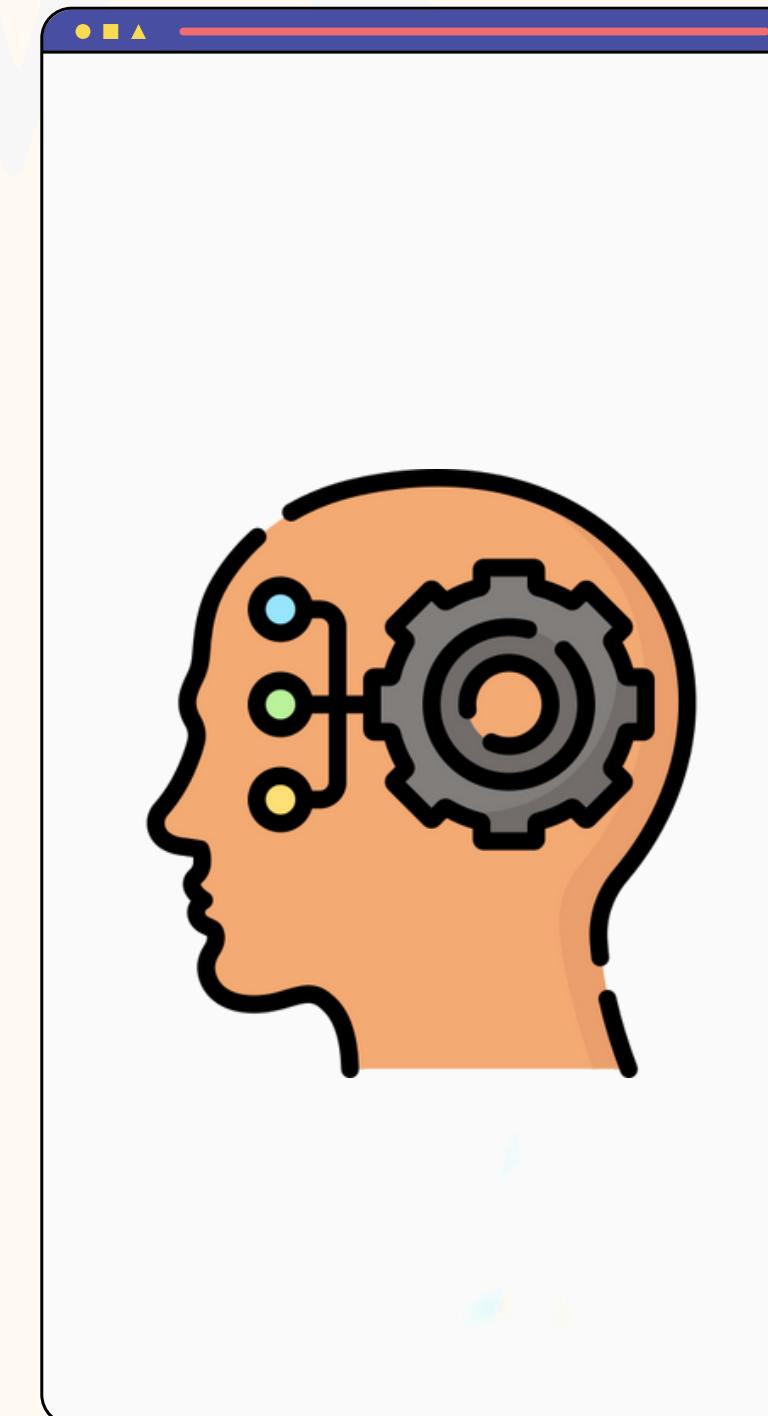
Resolver o jogo e tentar criar desenvolve competências



Pensamento-Lógico



Intuito de Pesquisa



Pensamento Crítico



Ensino Informática

Evita Alzheimer

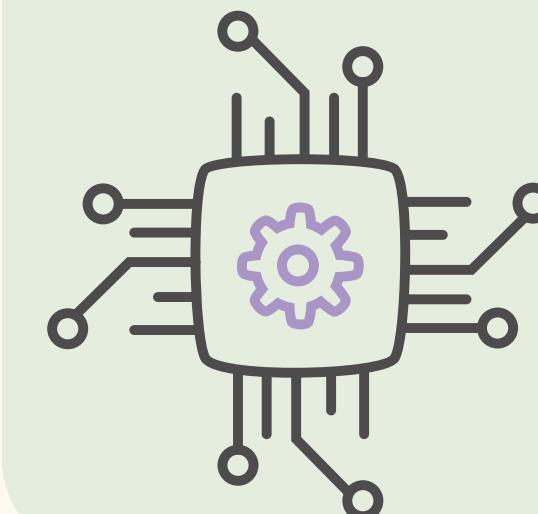
Um jogo de lógica que influencia os jogadores a construirem modelos de solução para tal situação do jogo. Dessa forma, eles constroem habilidades de resolução de problemas, explorar o raciocínio hipotético-dedutivo, generalizar soluções e procedimentos, observar regularidades.

Pesquisa mostra que atividades que estimulam o cérebro preservariam as estruturas vulneráveis e funções cognitivas em regiões cerebrais envolvidas na doença



Desafios Computacionais

A resolução eficiente do Jogo NIM apresenta desafios computacionais interessantes, incluindo a aplicação de algoritmos de busca e otimização. A análise computacional do jogo revela a interseção entre matemática e computação.



Estruturas

Relação jogo/código

```
struct jogo_informacao {
    int pontuacao;
    int *max_pacca;
    int *max_ratirar;
};

struct campeonato {
    int pontuacao_humano;
    int pontuacao_computador;
};
```

**Uso de Structs para a
criação e manipulação de
diferente tipos de dados,
sendo possível agrupar
mais de uma variável sob
o mesmo nome**



```
void inicializa_peca(estruct  
jogo_informacao *jogo_informacao){  
jogo_informacao->pontuacao = 0;  
jogo_informacao->max_peca = (int  
*)malloc(sizeof(int));  
jogo_informacao->max_rectilar = (int  
*)malloc(sizeof(int));  
}
```

**Uso e manipulação de
Ponteiros e Alocação de
Memória para inicialização
de valores que serão
alterados durante o
funcionamento do jogo**

```
    removez_jogador);
    if (resultado == 1) {
        printf("Fim do jogo! Você ganhou!\n");
        free(_informacao.mak_pecas);
        free(_informacao.mak_retirar);
        return 1;
    } else if (resultado == 0) {
        printf("Jogada inválida! Tente novamente.\n");
        continua;
    }
    printf("Você removeu %d peças!\n", removez_jogador);
    n -= removez_jogador;
    vez_de_pc = 1;
}

if (n > 0) {
    printf("Agora restam %d peças no tabuleiro.\n", n);
}
```

Uso do Free para liberar espaço de memória dos ponteiros anteriormente preenchidos pelo Malloc



```
void exibirJogo_Information(const  
struct jogo_Information *Information) {  
    //  
    int retirar_peca(struct  
    jogo_Information *Information, int  
    pecaTirada) {  
    //
```

**Passagem por referência
de estrutura para evitar a
cópia desnecessária de
dados e não comprometer
a eficiência do código**

```
do {  
    printf("\nBem-vindo ao jogo do NIM!\n");  
    printf("Escolha uma opção:\n");  
    printf("1. Jogar partida normal\n");  
    printf("2. Jogar campeonato\n");  
    printf("3. Salvar pontuação\n");  
    printf("4. Carregar pontuação\n");  
    printf("5. Sair\n");  
    scanf("%d", &escolha);  
    ...  
} while (escolha > 5 || escolha < 1);
```

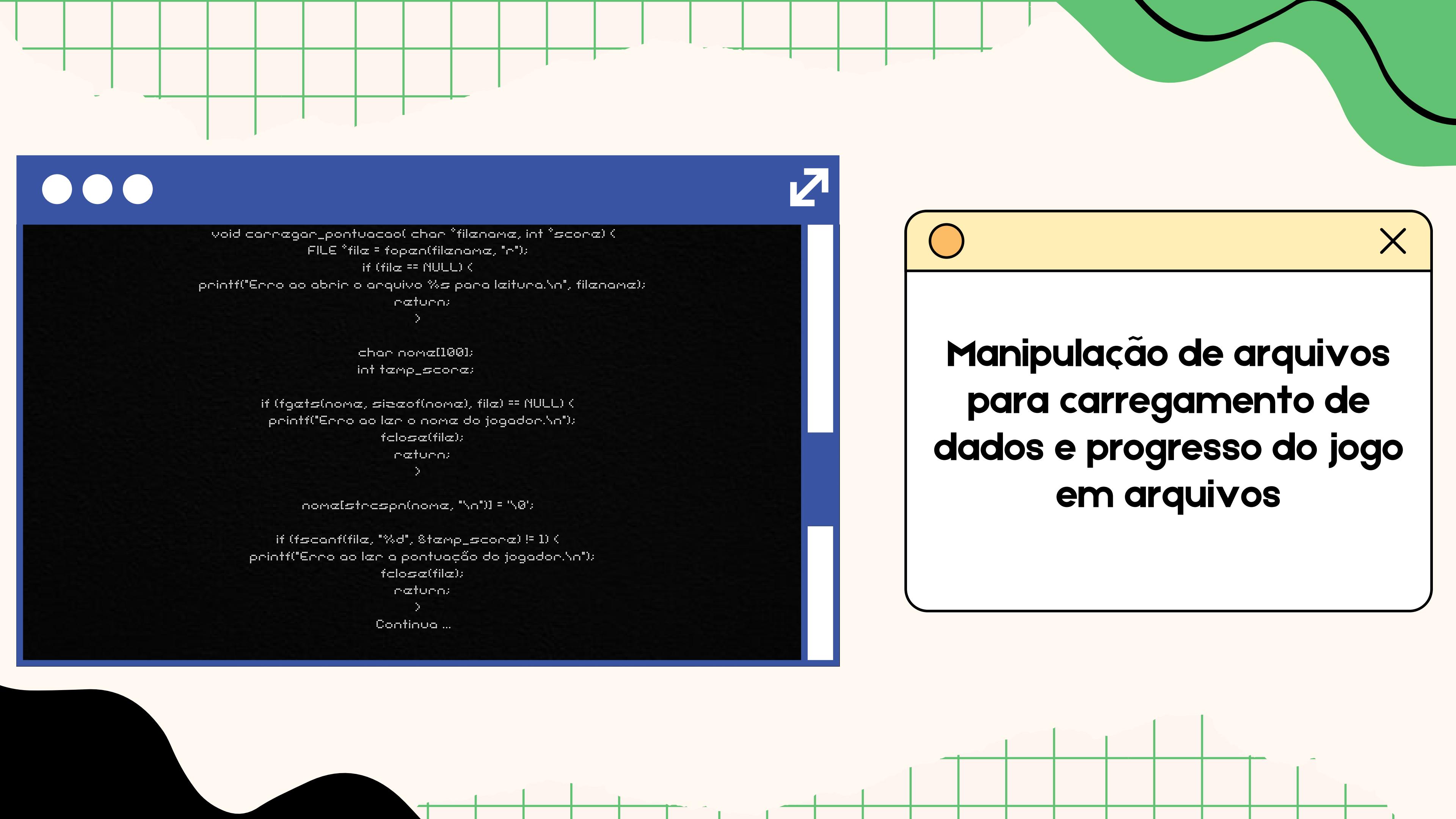
**Passagem por referência
de estrutura para evitar a
cópia desnecessária de
dados e não comprometer
a eficiência do código**

```
void exibir_jogo_informacao(const struct
    jogo_informacao *_informacao) {
    printf("Pontuação: %d\n", _informacao->pontuacao);
    if (_informacao->max_peca != NULL) {
        printf("Máximo das peças no tabuleiro: %d\n",
            *_informacao->max_peca);
    }
    if (_informacao->max_retirar != NULL) {
        printf("Máximo das peças a serem retiradas por
            jogada: %d\n",
            *_informacao->max_retirar);
    }
}
```

**Uso de vetores e string
para manipulação de dados
e/ou exibir e reter
informações pelo jogo**

```
void salvar_pontuacao(const char *filenamz, int  
                      *score) {  
    FILE *filez = fopen(filenamz, "w");  
    if (filez == NULL) {  
        printf("Erro ao abrir o arquivo para salvar.\n");  
        return;  
    }  
    fprintf(filez, "%s\n%d\n", filenamz, *score);  
    fclose(filez);  
    printf("Pontuação salva com sucesso.\nNome:  
          %s\nScore: %d\n", filenamz,  
                *score);  
}
```

Manipulação de arquivos para salvamento de dados e progresso do jogo em arquivos



Manipulação de arquivos para carregamento de dados e progresso do jogo em arquivos

```
void carregar_pontuacao( char *filename, int *score) {  
    FILE *file = fopen(filename, "r");  
    if (file == NULL) {  
        printf("Erro ao abrir o arquivo %s para leitura.\n", filename);  
        return;  
    }  
  
    char nome[100];  
    int temp_score;  
  
    if (fgetss(nome, sizeof(nome), file) == NULL) {  
        printf("Erro ao ler o nome do jogador.\n");  
        fclose(file);  
        return;  
    }  
  
    nome[sizeof(nome)-1] = '\0';  
  
    if (fscanf(file, "%d", &temp_score) != 1) {  
        printf("Erro ao ler a pontuação do jogador.\n");  
        fclose(file);  
        return;  
    }  
    *score = temp_score;  
    printf("Nome: %s\nPontuação: %d\n", nome, *score);  
    Continua ...
```

```
strcpy(filename, nome);
    *scorez = temp_scorez;

    fclose(file);
    printf("Pontuação carregada com sucesso.\n");
    printf("Nome do jogador: %s\n", nome);
    printf("Pontuação: %d\n", *scorez);
}

float campeonatos(int partidas, int *total_campzonatos_ganhos,
                   int *total_campzonatos_feitos) {
    float scorez_final = 0.0;
    for (int i = 0; i < partidas; i++) {
        if (partida(i)) {
            (*total_campzonatos_ganhos) += 1;
            (*total_campzonatos_feitos) += 1;
        } else {
            (*total_campzonatos_feitos) += 1;
        }
    }

    if (*total_campzonatos_feitos > 0) {
        scorez_final = ((float)(*total_campzonatos_ganhos) * partidas) /
                       *total_campzonatos_feitos;
        printf("%f", scorez_final);
    }
    return scorez_final;
}
```

**Continuação da função
carregar_pontuacao que é
responsável por carregar o
progresso do jogo**



Obrigado