RELATÓRIO TÉCNICO: Implementação do jogo “Ligue 4” em linguagem C

Universidade Federal de Sergipe (UFS)

Departamento de Computação

Disciplina: Programação Imperativa

Docente: Prof. Adam Lucas Pinheiro da Silva

Discentes:

- Arthur Nicolas Silva Lima
- Lauane de Morais Araujo
- Luiza Pereira Accioly

---

1. Introdução

O presente relatório descreve o desenvolvimento de uma aplicação digital do jogo de tabuleiro "Ligue 4" (Connect 4), implementada utilizando a linguagem de programação C.

O projeto foi requisito avaliativo da disciplina de Programação Imperativa, visando aplicar conceitos fundamentais da computação, como manipulação de matrizes, estruturas de controle, modularização de código, automação de compilação e boas práticas de desenvolvimento de software.

O objetivo do jogo consiste em alinhar quatro peças da mesma cor em um tabuleiro vertical de dimensões 6x7, alternando turnos entre dois jogadores (humanos ou artificiais). O desafio técnico concentrou-se na lógica de verificação de vitória em múltiplas direções, na implementação de uma inteligência artificial capaz de competir com o usuário e na representação gráfica do projeto.

---

2. Arquitetura e Estrutura do Projeto

Para garantir a manutenção e a escalabilidade do software, a equipe optou por uma abordagem modular, separando a interface, a lógica de jogo e as definições globais. O projeto segue uma estrutura de diretórios padrão:

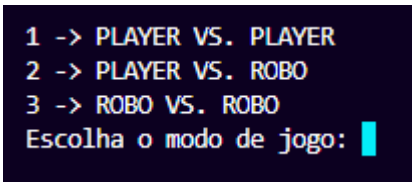
- **`src/` (Source):** Contém todos os arquivos de implementação (`.c`).
- **`include/` (Headers):** Contém os arquivos de cabeçalho (`.h`), que expõem as assinaturas das funções.

- **`Makefile`:** Arquivo de script para automação da compilação.

2.1 Módulos Desenvolvidos

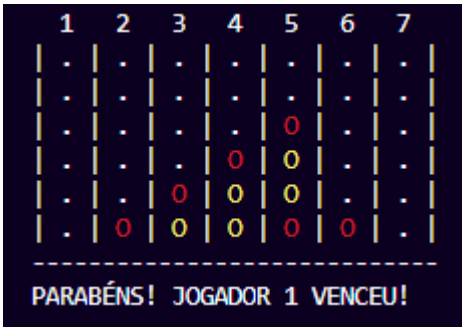
A aplicação foi dividida nos seguintes componentes lógicos:

1. **`main.c`:** Módulo principal responsável pelo fluxo de execução, gerenciamento do *loop* do jogo, alternância de turnos e chamadas às funções lógicas.
2. **`display.c` / `.h`:** Responsável pela camada de apresentação no terminal. Utiliza sequências de caracteres ANSI para renderizar o tabuleiro colorido e limpar a tela, melhorando a experiência do usuário.
3. **`verificarcolunas.c` / `.h`:** Implementa a física do jogo. Este módulo gerencia a gravidade das peças, percorrendo a matriz da base para o topo para encontrar a primeira posição disponível em uma coluna.
4. **`vitoria.c` / `.h`:** Contém os algoritmos de verificação de estado final. Analisa a matriz após cada jogada para detectar sequências de quatro peças iguais.
5. **`robotnivel1.c` / `.h`:** Implementa a Inteligência Artificial de nível básico, baseada em geração pseudo aleatória de jogadas.



Menu inicial do jogo.

Tabuleiro do jogo.



Tela de vitória.

---

3. Algoritmos e Lógica de Implementação

A lógica central do jogo baseia-se na manipulação de uma matriz de inteiros `int tabuleiro[6][7]`, onde `0` representa espaço vazio, `1` representa o Jogador 1 (Vermelho) e `2` representa o Jogador 2/Robô (Amarelo).

3.1 Mecânica da Gravidade

O Ligue 4 exige que a peça ocupe a posição mais baixa disponível na coluna selecionada. A função `verificarcolunas` recebe a coluna escolhida pelo usuário e itera sobre as linhas de baixo para cima (índice 5 ao 0):

```
C
for(int k=5; k>=0; k--){
   if(tabuleiro[k][coluna] == 0){
      tabuleiro[k][coluna] = jogador;
      break; // Interrompe assim que a peça "pousa"
   }
}
```

3.2 Detecção de Vitória

A detecção de vitória é o algoritmo de maior complexidade do projeto. A função `vitoria` verifica quatro condições distintas a cada nova peça inserida:

1. **Horizontal:** Verifica se `matriz[i][j]` é igual aos seus 3 sucessores na mesma linha (`j+1`, `j+2`, `j+3`).
2. **Vertical:** Verifica se `matriz[i][j]` é igual aos seus 3 sucessores na mesma coluna.
3. **Diagonais:**
    - *Diagonal Principal:* Verifica a sequência descendente (`i+1`, `j+1`).
    - *Diagonal Secundária:* Verifica a sequência ascendente (`i-1`, `j+1`).

Foi implementada uma validação rigorosa dos índices dos laços de repetição (`for`) para evitar *Segmentation Fault*, garantindo que o algoritmo nunca tente acessar posições de memória fora dos limites da matriz (ex: tentar verificar a coluna 8 em uma matriz de tamanho 7).

3.3 Inteligência Artificial (Nível 1)

O oponente computacional utiliza a biblioteca `<stdlib.h>` e `<time.h>`. O algoritmo `robotnivel1` gera um número aleatório entre 1 e 7. Caso a coluna sorteada esteja cheia, o sistema realiza novos sorteios recursivos ou iterativos até encontrar uma jogada válida, garantindo que o robô nunca perca a vez.

---

4. Compilação e Execução

Para otimizar o processo de desenvolvimento e avaliação, foi criado um **Makefile**. Isso permite a compilação de múltiplos arquivos com dependências cruzadas através de um único comando no terminal Linux/Unix.

- **Comando de compilação:** `make` (Gera o executável `jogo` linkando todas as bibliotecas).
- **Limpeza de arquivos:** `make clean` (Remove arquivos binários temporários).

---

5. Distribuição de Tarefas

O desenvolvimento foi realizado de forma colaborativa, com a seguinte divisão de responsabilidades:

- **Arthur Nicolas:**
- **Lauane Morais:**
- **Luiza Accioly:**

---

6. Conclusão e Dificuldades Encontradas

O desenvolvimento do projeto Ligue 4 permitiu a consolidação dos conhecimentos em lógica de programação e estruturação de dados em C.

A principal dificuldade encontrada pela equipe foi a lógica das diagonais, que exigiu testes manuais extensivos para garantir que as extremidades do tabuleiro não gerassem falsos positivos ou erros de acesso à memória.

Outro desafio foi a **modularização correta do código**. Inicialmente, houve problemas de linker (erros de "multiple definition") ao tentar compartilhar variáveis globais entre arquivos. A solução foi adotar o uso de arquivos de cabeçalho (`.h`) apenas para declarações de funções e constantes, mantendo a implementação nos arquivos `.c`.

O resultado final é um software que atende a todos os requisitos funcionais propostos, apresentando uma interface limpa e uma jogabilidade fluida no terminal.

Esperamos que goste!