Relatorio do projeto de Tecnicas de Desenvolvimento e Algoritmo

Integrantes:

- João Pedro Silva Nobre

RGM: 42976154

- Ismael Alves Martins da Silva

RGM: 42330254

- Ian Rodrigues Simões Silva

RGM: 42942471

- José Caio Linhares Rodrigues Monteiro

RGM: 42623201

- Antonio Arthur Nunes da Silva

RGM: 43768318

Introdução:

Este software consiste em uma simulação digital do clássico jogo de tabuleiro "Pula Pirata", desenvolvida em linguagem C para execução via terminal (console). O jogo foi projetado para **dois jogadores** que alternam turnos utilizando a mesma máquina (modo *hotseat*). A interface gráfica é representada por caracteres ASCII, onde um "barril" virtual possui 9 orifícios numerados disponíveis para seleção.

Regras:

Entradas Válidas: O sistema aceita exclusivamente números inteiros no intervalo de 1 a 9, correspondentes aos orifícios do barril. Tratamento de Exceções: O sistema possui uma validação robusta de tipos de dados. Caso o jogador insira: O dígito 0; Letras (a-z, A-Z); Caracteres especiais ou símbolos; O jogo rejeitará a entrada imediatamente e exibirá a mensagem de erro: "Opção Inválida", solicitando uma nova entrada correta antes de prosseguir para o próximo turno ou jogador.

Resultados:

O objetivo é selecionar orifícios (números) em um "barril" virtual sem acionar a posição que faz o pirata pular. O barril é ilustrado através de arte ASCII, e o sistema utiliza geração de números pseudoaleatórios para definir a posição "fatal" a cada nova partida, garantindo que o jogo nunca seja previsível.

Documentação do Projeto: Pula Pirata em C

1. Descrição Geral do Jogo

O projeto consiste em uma implementação do jogo "Pula Pirata" em linguagem C, executado via console (terminal). O jogo foi projetado para dois jogadores que alternam turnos na mesma máquina (*hotseat*). O objetivo é selecionar orifícios (números) em um "barril" virtual sem acionar a posição que faz o pirata pular.

O barril é ilustrado através de arte ASCII, e o sistema utiliza geração de números pseudoaleatórios para definir a posição "fatal" a cada nova partida, garantindo que o jogo nunca seja previsível.

Exemplificação do Código Fonte

Abaixo, destaca-se o trecho responsável pela inicialização da aleatoriedade e definição da posição fatal ("bomba"), garantindo que a cada execução a posição perdedora mude:

C

```
srand(time(NULL)); // Inicializa a semente do RNG com o tempo atual


// Sorteia a posição fatal entre 0 e 9
fatal = rand() % 10;
espadas[fatal] = 1; // Define o valor 1 como a posição da bomba
```

- Também destaca-se a lógica de alternância de turnos entre o Jogador 1 e o Jogador 2:

C

```
// Alternância simples de turnos
if (turnos[turno_atual] == 1) {
    printf("\nJOGADOR 1:");
} else {
    printf("\nJOGADOR 2:");
}
// ... Lógica do jogo ...
turno_atual++;
if (turno_atual == 2) {
    turno_atual = 0; // Reseta para o primeiro jogador
}
```

2. Dificuldades Encontradas e Soluções Implementadas

Durante o desenvolvimento, foram identificados desafios técnicos relacionados à interação do usuário com o terminal e à lógica de vetores.

A. Tratamento de "Lixo" no Buffer de Entrada

Dificuldade: Ao digitar uma letra ou símbolo quando o programa esperava um número (via `scanf`), o programa entrava em um *loop infinito*. Isso ocorria porque o caractere inválido permanecia no buffer de entrada e o `scanf` tentava lê-lo repetidamente sem sucesso.

Solução: Foi implementada a função auxiliar `limparBuffer()`. Esta função consome todos os caracteres restantes no buffer de entrada até encontrar uma quebra de linha, limpando a sujeira antes da próxima leitura.

C

```
void limparBuffer() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF) {}
}
```

B. Mapeamento de Entrada (Usuário vs. Array)

Dificuldade: Para o usuário, é intuitivo escolher números de **1 a 10**. No entanto, vetores em C são indexados de **0 a 9**. Usar a entrada direta causaria erros de acesso à memória (segmentation fault) ou lógica incorreta.

Solução: Foi implementado um decremento imediato após a leitura da entrada do usuário, ajustando o valor para o índice correto do array antes de qualquer verificação.

C

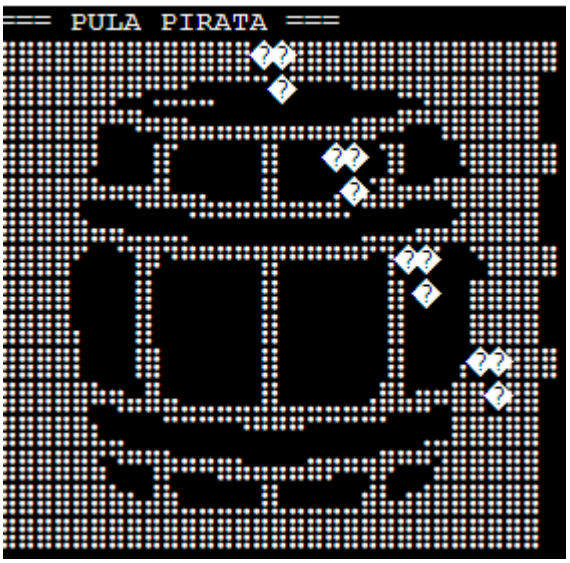
```
scanf("%d", &escolha);
escolha--; // Ajusta input 1-10 para índice 0-9
```

3. Demonstrativo das Funcionalidades e Algoritmo em Funcionamento

Abaixo segue o registro do algoritmo em execução, demonstrando desde a inicialização até o encerramento da partida.

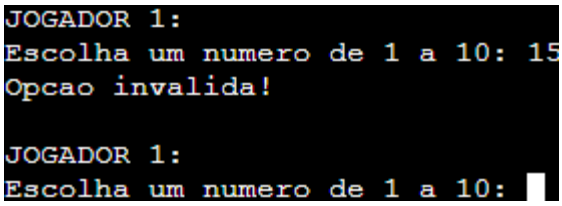
Passo 1: Inicialização e Interface

Ao iniciar, o sistema exibe a arte ASCII do barril e solicita a entrada do Jogador 1. O sistema valida internamente a posição fatal (invisível ao usuário).



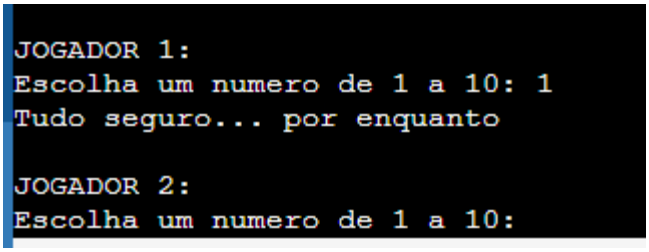
Passo 2: Validação de Entradas Inválidas

Se o usuário digitar um número fora do intervalo (ex: 15 ou -5) ou caracteres inválidos, o sistema aciona o tratamento de erro, limpa o buffer e pede o número novamente sem trocar de turno.



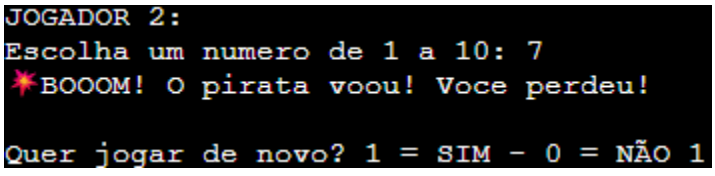
Passo 3: Jogada Segura e Troca de Turno

Quando um jogador escolhe um número que não é a bomba, o sistema confirma que está "Tudo seguro" e passa a vez imediatamente para o próximo jogador.



Passo 4: Condição de Derrota e Replay

Quando a posição `espadas[escolha] == 1` é selecionada, o laço `while` é interrompido. O sistema exibe a mensagem de derrota ("BOOOM") e pergunta se os usuários desejam jogar novamente.



Apêndice

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void limparBuffer() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF) {}
}

int jogo() {
    int espadas[10] = {0}; // 0 = seguro, 1 = bomba
    int fatal, escolha;
    int turnos[2] ={1,2};
    int turno_atual=0;

    srand(time(NULL)); // inicia o RNG

    // Sorteia a posição fatal
    fatal = rand() % 10;
    espadas[fatal] = 1;

    printf("=== PULA PIRATA ===\n");
    printf(
"⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⣿⣿⡿⠿⢛⣛⣋⣉⠉⠉⠉⠉⠉⠉⠉⠙⠛⠛⠿⢿⣿⣿⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⣿⡟⠛⠻⠶⣶⣦⣤⣤⣤⣤⣤⣤⣤⣤⣴⣶⠶⠟⠛⢻⣿⣿⣿⣿⣿\n"
```

```
"⣿⣿⣿⣿⣿⠇⠀⠀⢸⡏⠀⠀⠀⠀⢸⡇⠀⠀⠀⠀⢹⡇⠀⠀⠸⣿⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⡿⠷⠶⢲⣾⣧⣤⣄⣀⣀⣸⣇⣀⣀⣠⣤⣬⣿⡶⠶⠿⣿⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⣷⣤⣤⣀⣀⣀⡈⠉⠉⠉⠉⠉⠉⠉⠉⠁⣀⣀⣀⣤⣤⣾⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⠋⠀⠈⢹⡿⠙⠛⠛⠛⠛⢻⡟⠛⠛⠛⠛⠋⢹⡏⠁⠀⠙⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⠀⠀⠀⢸⡇⠀⠀⠀⠀⠀⢸⡇⠀⠀⠀⠀⠀⢸⡇⠀⠀⠀⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⡄⠀⠀⢸⡇⠀⠀⠀⠀⠀⢸⡇⠀⠀⠀⠀⠀⢸⡇⠀⠀⠀⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⡇⠀⠀⢸⣿⠀⠀⠀⠀⠀⢸⡇⠀⠀⠀⠀⠀⢸⡇⠀⠀⢠⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⣿⠷⢖⣤⣿⣤⣀⣀⣀⣀⣸⣇⣀⣀⣀⣀⣠⣿⣧⣶⠶⣿⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⣿⣆⣀⠀⠀⠉⠉⠉⠉⠙⠛⠛⠛⠉⠉⠉⠉⠁⠀⢀⣀⣿⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⣿⣯⠙⠛⢻⡷⠶⠶⣶⣦⣤⣤⣤⣶⡶⠶⢶⡿⠛⠛⣽⣿⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⣿⣿⣷⣦⣼⣷⣀⣀⣀⣀⣸⣇⣀⣀⣀⣀⣾⣧⣴⣾⣿⣿⣿⣿⣿⣿\n"
"⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿\n"
  );

  while (1) {
    if (turnos[turno_atual]==1)
    {
      printf("\nJOGADOR 1:");
    }else{
      printf("\nJOGADOR 2:");
    }
    printf("\nEscolha um numero de 1 a 10: ");
    scanf("%d", &escolha);



    // Ajusta para índice do array (0 a 9)
    escolha--;

    // Verifica entrada válida
    if (escolha < 0 || escolha > 9) {
```

```
            printf("Opcao invalida!\n");
            limparBuffer();
            continue;
        }

        // Verifica se caiu na posição fatal
        if (espadas[escolha] == 1) {
            printf(" 💥 BOOOM! O pirata voou! Voce perdeu!\n");
            break;
        } else {
            printf("Tudo seguro... por enquanto\n");
            turno_atual++;
            if (turno_atual==2){
                turno_atual=0;
            }
        }
    }
    return 0;

}




int main() {
    int jogar;

    do {
        jogo();
```

```
    printf("\nQuer jogar de novo? 1 = SIM - 0 = NÃO ");
    scanf("%d", &jogar);

} while (jogar == 1);

printf("Até mais! \n");
return 0;
}
```