RELATÓRIO DO TRABALHO DE PIF

Professor: João Victor Tinoco de Souza Abreu
Alunos: Lucas Chaves de Albuquerque, Luiz Gustavo Gonçalves da Silva, Heitor Castilhos de Melo

1. Introdução

Este relatório apresenta, de forma resumida, o desenvolvimento de um jogo de **Batalha Naval em linguagem C**, realizado como parte do Projeto Integrador Final (PIF) do curso de Sistemas de Informação da CESAR School.

O objetivo do projeto foi colocar em prática conceitos importantes de programação em C, como **structs, enums, ponteiros, alocação dinâmica de memória, organização do código em múltiplos arquivos** (.h e .c) e trabalho em equipe com divisão de módulos.

---

2. Arquitetura Geral do Sistema

O sistema foi dividido em módulos, cada um com uma responsabilidade clara:

2.1. Módulo do Tabuleiro (`board.h` / `board.c`)

Define:

- o `enum EstadoCelula` (água, navio, acerto, erro);
- a `struct Celula`, com o estado e o `id_navio`;
- a `struct Tabuleiro`, com linhas, colunas e um vetor dinâmico de células.

Implementa:

- `inicializarTabuleiro` (aloca memória e inicializa tudo como água);
- `liberarTabuleiro` (libera memória e zera o tabuleiro);
- `obterCelula` (converte linha/coluna em índice e retorna a célula certa);
- `podeColocarNavio` e `colocarNavio` (validação e posicionamento de navios, respeitando limites e evitando colisão).

2.2. Módulo da Frota (`fleet.h` / `fleet.c`)

Define as structs:

- `Navio` (nome, tamanho, acertos);
- `Frota` (vetor dinâmico de navios + quantidade).

Implementa:

- `inicializarFrota`, `adicionarNavioNaFrota` com `realloc`, `criarFrotaPadrao`, `liberarFrota` e `frotaAfundou` (que verifica se todos os navios já foram completamente acertados).

2.3. Módulo do Jogo (`game.h` / `game.c`)

Define:

- `Jogador` (tabuleiro de navios, tabuleiro de tiros, frota, apelido, estatísticas);
- `Jogo` (dois jogadores, controle de turno, flag de fim de jogo, configuração de tamanho).

Implementa:

- `inicializarJogador` e `liberarJogador`;
- `posicionandoFrotaAuto` (usa números aleatórios para posicionar navios válidos);
- `posicionandoFrotaManual` (permite o jogador escolher coordenadas e orientação, com validação);
- `jogadorAtira`, que cuida de toda a lógica de um tiro, atualização de estados no tabuleiro e contagem de acertos dos navios.

2.4. Entrada/Saída e Funções Auxiliares (`io`, `rnd`, `main`, `makefile`)

`io.h` / `io.c`:

- `exibirTabuleiro` desenha o tabuleiro usando letras (linhas), números (colunas) e símbolos (`X`, `.`, `N`, `~`);

- `lerCoordenada` converte entrada do tipo `"A1"` para índices de linha/coluna;
- `exibirMensagem` imprime mensagens simples.

`rnd.h` / `rnd.c`:

- `inicializarAleatorio` configura a semente do `rand()` com `time(NULL)`;
- `obterNumeroAleatorio` auxilia na geração de números entre `min` e `max`, usado para posições aleatórias.

`main.c`:

- ponto de entrada do programa;
- inicializa o sistema (seed aleatória, jogadores, tabuleiros, frotas);
- controla o loop principal do jogo até que uma das frotas afunde;
- chama as funções de liberação de memória ao final.

`Makefile`:

- arquivo de configuração que automatiza o processo de compilação, gerando o executável do jogo a partir dos arquivos `.c`.

---

3. Funcionamento Geral do Jogo

O fluxo básico do jogo é:

1. O programa é iniciado (`main.c`), o gerador de números aleatórios é configurado e os jogadores são inicializados.
2. Os tabuleiros são criados dinamicamente com o tamanho configurado, e as frotas são montadas com navios de tamanhos diferentes.
3. Os navios podem ser posicionados automaticamente (modo aleatório) ou manualmente (jogador escolhe coordenadas e orientação), sempre respeitando regras de limite e colisão.

4. Durante o jogo, os jogadores se alternam atirando em coordenadas do tabuleiro do adversário:
    - se a célula contém navio, é marcado acerto;
    - se for água, é marcado erro;
    - o sistema controla quantos acertos cada navio levou até afundar.
5. Quando todos os navios de uma frota atingem seu limite de acertos, a função `frotaAfundou` identifica que o jogo acabou e o vencedor é determinado.
6. Ao final, toda a memória dinâmica alocada (tabuleiros e frotas) é liberada para evitar vazamento de memória.

---

4. Testes e Dificuldades

Durante o desenvolvimento, foram realizados testes criando tabuleiros com tamanhos diferentes, posicionando navios em bordas e tentando sobrepor navios para verificar se as validações estavam corretas. Também foram testadas leituras de coordenadas inválidas (como letras fora do intervalo e números maiores que o tamanho do tabuleiro) e repetição de tiros na mesma posição.

As principais dificuldades foram entender o uso de ponteiros com structs, a conversão de matriz 2D para vetor 1D e o controle da memória dinâmica com `malloc`, `realloc` e `free`, mas essas partes foram compreendidas à medida que o grupo foi analisando e testando o código em conjunto.

5. Conclusão

O desenvolvimento deste projeto de Batalha Naval em C permitiu ao grupo praticar, na prática, conceitos que normalmente são mais difíceis apenas na teoria: ponteiros, alocação dinâmica, structs, enums, modularização do código e uso de múltiplos arquivos.

A divisão de responsabilidades entre os diferentes módulos do sistema (tabuleiro e frota, lógica do jogo, entrada/saída, aleatoriedade, função principal e Makefile) ajudou a organizar o trabalho e aproximou o projeto de um cenário real de desenvolvimento em equipe.

Além de cumprir os requisitos da disciplina, o projeto contribuiu para que o grupo ganhasse mais confiança em C e na leitura e compreensão de códigos maiores e mais estruturados.