Relatório Trabalho 1 de Intro C/C++ 2020-2

Batalha Naval

Aluno: Helena Coelho Nicolli DRE: 120016218
Aluno: Luiz Carlos Ferreira Carvalho DRE: 120025788
Aluno: Yuri Ferreira Melo DRE: 120081378

A. Descrição do Jogo e seu Objetivo

O jogo desenvolvido foi o Batalha Naval, que possui 3 modalidades, a primeira onde o jogador enfrenta o computador, com 3 opções de dificuldade, a segunda onde 2 jogadores se enfrentam, e a terceira onde dois computadores se enfrentam. Em todos os modos o objetivo é afundar todos os navios do inimigo alcançando a pontuação máxima de 14 pontos (Soma dos tamanhos dos 4 navios, Submarino, Destroyer, Cruzado e Porta-Aviões).

Ele consiste em escolher posições para os navios em um tabuleiro 10x10, sempre marcando a primeira e última posições desejadas para cada navio. Os jogadores podem pedir para o computador preencher o tabuleiro automaticamente em vez de eles mesmos o preencherem manualmente.

B. Regras do Jogo

I - Cada jogador deve posicionar os navios em seu tabuleiro.

II - Cada navio tem um tamanho fixo.

III - Os navios não podem se sobrepor.

IV - Os jogadores não podem ver o tabuleiro um do outro.

V - O jogador deve escolher uma posição para atacar.

VI - O jogador alvo deve dizer se o outro acertou alguma parte do navio.

VII - Se o jogador acertar ele joga novamente até errar.

VIII - Se o jogador acertar todas as partes de um navio o outro deve avisar que esse afundou.

IX - Ganha quem afunda todos os navios.

C. Organização do Código

O código implementa 3 estruturas, a primeira definida como “Jogo” que contém as principais informações sobre o jogo, como dificuldade e modalidade escolhidas, a segunda estrutura “Jogador” contém as informações sobre cada participante do jogo, se ele é um usuário ou computador, armazena o tabuleiro, o nome, a pontuação, se acertou a jogada anterior entre outros, a terceira estrutura Navio armazena o nome, a classe, e o tamanho de cada embarcação.

A utilização das estruturas foi extremamente útil ao longo de todo projeto, pois elas permitem que alterações feitas em variáveis dentro de uma função fiquem salvas de forma global ao longo da execução do código, além de ser fácil de armazenar dados para cada jogador, como por exemplo saber se o usuário acertou o disparo anterior ou qual foi a jogada anterior dele, diferenciando a rotina do programa a partir desses dados.

Ele foi construído por partes, sempre pensando em etapa por etapa do jogo, as primeiras parte feitas foram a de criação dos jogadores, do tabuleiro e entrada das posições dos navios, que demandou uma função bastante complexa para verificar se a posição escolhida é válida ou não. O código também foi pensado para ser reutilizável em outras partes, então funções como imprimeTabuleiro foram criadas para evitar repetições, além de permitir que uma função setTabuleiro (fazia parte anteriormente de setNavio antes de ser dividida) fosse usada tanto pelo usuário, tanto pelo computador, com poucas funções sendo feitas exclusivamente para uma situação como a realizaDisparoIA que é bastante diferente da realizaDisparo dos jogadores.

O código foi pensado originalmente com o modo Jogador vs CPU como base, os modos de 2 jogadores e CPU x CPU foram adicionados posteriormente, este último adicionado pois como a rotina da CPU contra o jogador estava pronta, nada impedia de colocar duas CPU se enfrentando.

A existência de níveis de dificuldade foi planejada desde o início e implementada na função realizaDisparoIA com 3 possibilidades, a primeira fácil onde a CPU sempre faz disparos aleatórios, a segunda normal, onde a CPU após um acerto tenta o disparo em uma posição em volta do acerto, e a terceira difícil, onde a CPU após o primeiro acerto sempre acerta as próximas jogadas até o fim do navio, podendo não afunda-lo caso o primeiro acerto tenha sido no meio.

Para a parte do código de sortear as posições tanto para setTabuleiro quanto realizaDisparo foi usada a função rand() que demandou bastante atenção para ser usada corretamente e não repetir sequências semelhantes, por exemplo quando usada na função sorteiaTitulo ambos os jogadores acabavam recebendo o mesmo título de forma seguida pois o tempo entre a definição do título de um e do outro não era grande o bastante, por isso passou-se a se utilizar como semente na srand() além da “hora” da função time() o tamanho de cada nome com a strlen().

Rotina do código:

1 - Função main inicia e cria os 4 tipos de Navios com a função setNavio, passando os atributos de cada um como tamanho, nome e classe.

2 - Inicia a função setJogo que a partir das funções chamadas dentro dela (escolheDificuldade e escolheModoJogo) define como será a partida e retorna o jogo para a variável do tipo jogo iniciada na main.

3 - A partir do modo de jogo escolhido entra em um dos ifs definidos para diferenciar o que fazer em cada modo.

4 - Chama a função setJogador que retorna uma variável do tipo Jogador aonde constam os dados de cada jogador como nome, título (definido na função sorteiaTitulo), se é humano ou não etc. Além de preencher o tabuleiro do jogador com água (‘~’) (função preencheTabuleiro) e a lista de jogadas realizadas com n (função preencheJogadas) para eliminar qualquer lixo, e poder fazer comparações mais tarde.

5 - No caso de um usuário inicia uma rotina para saber se ele quer colocar os navios no tabuleiro de forma manual ou automatizada pela CPU. Os navios sempre são colocados a partir da função setTabuleiro que determina o que fazer a partir dos atributos de entrada, como modo de jogo entre outros, além de receber a variável do tipo navio, que será colocado no tabuleiro.

6 - Dentro da setTabuleiro é chamada a função getPosicao responsável por ler as entradas do usuário no caso de escolha manual das posições dos navios, ou usar as funções sorteiaI e sorteiaJ para posições automatizadas. São sempre escolhidas ou escolhidas as posições Inicial e Final do navio.

7 - Após isso, ainda dentro da setTabuleiro é chamada a função verificaPosicao, para checar se as posições escolhidas ou sorteadas anteriormente são válidas, e então colocar o navio no tabuleiro. também é feita a diferenciação para quando é uma posição inicial e quando é final.

8 - A rotina de ler entrada, verificá-la e colocar no tabuleiro é feita para cada jogador independente do tipo, quando o modo de jogo envolve um usuário ele pode escolher se está satisfeito ou se deseja escolher as posições novamente de todos os navios.

9 - Após a preparação para o jogo terminar, ele entra em um while que repete a chamada das funções de disparo para cada jogador.

10 - Se for um jogador usuário ele inicia a função realizaDisparo, que verifica se a jogada já foi realizada antes e se está dentro do tabuleiro, após isso verifica se acertou, somando 1 ponto em caso de acerto e alterando o atributo acertou disparo anterior para true.

11 - Se o jogador for computador ele chama a função realizaDisparoIA que se baseia na dificuldade escolhida na setJogo para determinar o comportamento do disparo da IA, que no caso fácil, sempre sorteia o disparo, no normal após primeiro acerto tenta acertar em volta do primeiro, e no difícil sempre acerta o segundo tiro seguindo em umas das 4 posições possíveis a partir do primeiro tiro, podendo afundar completamente ou não um navio.

12 - No caso de acertar o disparo a função entra em outro while que se repetirá enquanto o jogador acertar a jogada, pois de acordo com as regras do jogo, sempre que acerta um navio o jogador jogo de novo.

13 - Em casos de acertos a função verificaSeAfundou é chamada para verificar se o navio atingido afundou ou não. Além de sempre salvar a posição escolhida para disparo na lista de jogadas realizadas do jogador ou computador mesmo em caso de erro.

14 - Caso o jogador erre o tiro ou atinja a pontuação máxima ele sai do while e passa para a verificação que encerra o jogo quando um jogador atinge a pontuação máxima.

15 - Caso o jogador não tenha atingido a pontuação máxima e tenha errado o disparo passa a vez do próximo jogador que repete a rotina de disparo descrita acima.15 - Se nenhum dos dois atinge a pontuação máxima e erra os disparos as rotinas são repetidas até que a pontuação máxima seja alcançada por alguma dos dois. A pontuação máxima é a quantidade de acertos necessários para afundar todos os navios de um jogador.

D. Dificuldades no desenvolvimento do Trabalho

Uma das maiores dificuldades encontradas foi a conexão de internet dos integrantes do grupo, pois a ideia original era trabalhar usando o LiveShare do Visual Studio Code.

Em relação ao código, a maior dificuldade foi a verificação de posição válida, pois existiam muitos casos a serem cobertos, e demandou um dia inteiro para ficar pronta. No início o grupo pensou em ler posição por posição para cada parte do Navio, por exemplo se o Cruzador tendo tamanho 4 seriam lidas 4 entradas. Porém optou-se por ler apenas duas entradas, uma representando a posição inicial e a outra a posição final, no caso do Cruzador com posição inicial A0 as posições finais possíveis seriam: A3 e D0.

Criar uma função que sorteia números também foi difícil pelas limitações da função rand() que, se usada de forma simplória, sempre sorteia o mesmo número.

Criar a modalidade para 2 jogadores também demandou tempo e paciência pois o grupo nunca chegava a um consenso sobre a exposição dos tabuleiros na tela dos jogadores, que, em teoria, estariam utilizando a mesma máquina para jogar.

E. Bibliotecas extras não ensinadas no período e utilizadas no código

Foram utilizadas as bibliotecas: windows.h para colocar cor no terminal do Windows, e time.h para usar de semente da função rand().

F. Aprendizados no desenvolvimento do jogo

Yuri: Aperfeiçoamento do uso e construção de matrizes em C.

Helena: Desenvolvimento de código seguindo uma lógica compartilhada e combinada pelo grupo, não individual. Implementação de modalidades e níveis de dificuldade a um jogo; uso de structs, pois exigem uma visão inicial abrangente do que o programa será e precisará posteriormente; implementar cor de texto, uso de tempo, função de seleção aleatória, em C.

Luiz: Aprender a sortear números pseudo-aleatórios em C, colocar cor nos terminais Unix e do Windows, melhorar o uso de vetores, strings e passagem de parâmetros em funções em C.

G. Códigos que foram utilizados de Inspiração ou Referência

Link e código consultado/usado para colocar cor no terminal do Windows:
https://www.programmersought.com/article/11014261013/

Links e códigos consultados/usados para colocar cor no terminal do Linux/Mac:
https://www.theurbanpenguin.com/4184-2/
https://www.programmersought.com/article/17911867710/

Link e código usado como referência para sorteio de números pseudo-aleatórios.
https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/

H. Sistema Operacional Utilizado

Luiz trabalhou no macOS, enquanto Yuri usou Linux e Helena trabalhou no Windows.