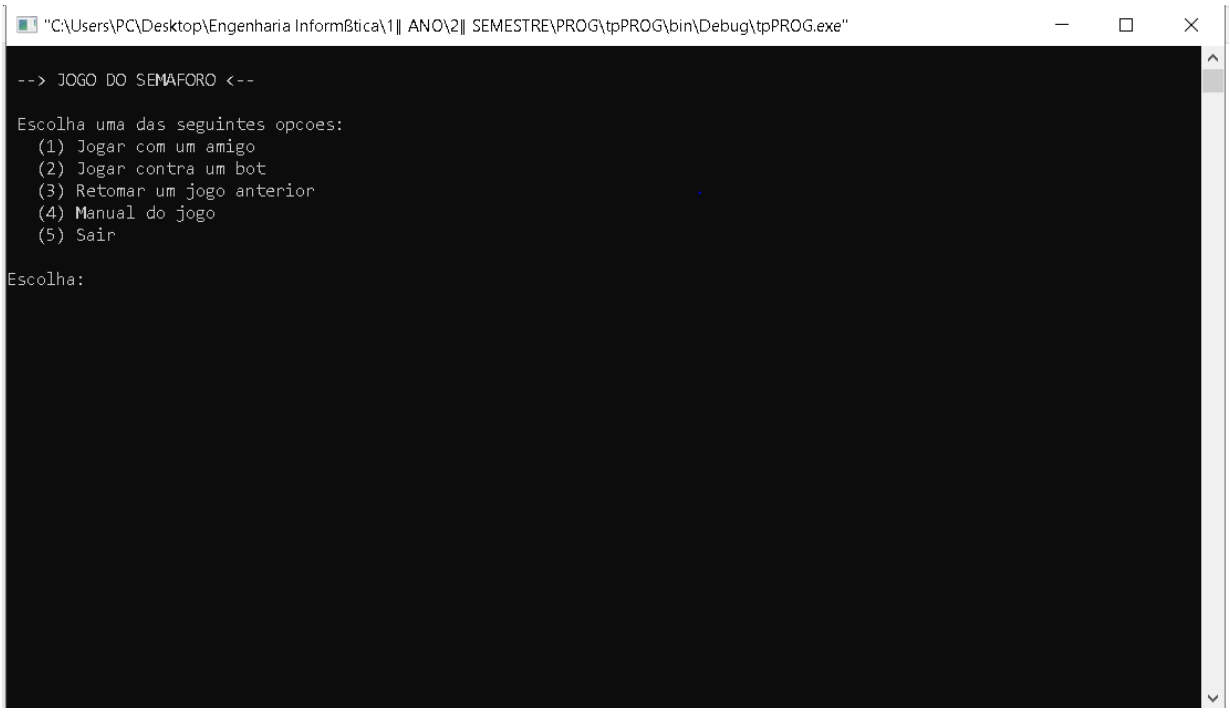
Relatório Trabalho Prático

ENGENHARIA INFORMÁTICA



Trabalho Realizado Por:

- Eduardo Correia 2020139576

Índice

Conteúdo

1.Introdução 3

2.Principais estruturas de dados 3

3.Estruturas dinâmicas implementadas 4

4.Opções tomadas em termos de implementação 5

5.Manual de utilização 8

6.Conclusão.....9

1. Introdução

Este trabalho foi realizado no âmbito da Unidade Curricular de Programação como trabalho prático. Com esta atividade adquiri e consolidei bastante os meus conhecimentos sobre a linguagem de programação em C, metendo os em prática com a criação deste jogo que nos foi pedido.

Este trabalho tinha como objetivo principal que cada aluno criasse o seu jogo do semaforo, com algumas implementacões extra, fora da normalidade do jogo original.

Realizei este trabalho na plataforma codeblocs.

2.Principais estruturas de dados

```
typedef struct jogador Jogador;

struct jogador{
    char nome[50];
    int linhas_colunas;
    int pedras;
};
```

A estrutura “Jogador” foi a estrutura que utilizei para guardar as informacoes sobre o nome dos jogadores, e sobre as linhas, colunas e pedras adicionadas por cada jogador, auxiliando durante o jogo a verificar se a jogada que o jogador A ou B deseja realizar ainda é possivel.

3.Estruturas dinâmicas implementadas

```
typedef struct lista
{
    char **jogos;
    int linhasjogos;
    int colunasjogos;
    int jogada;
    int linha;
    int coluna;
    struct lista* prox;
} Lista;
```

Esta estrutura “Lista” é uma variante de lista ligada, em que nela guardo cada estado do meu tabuleiro, assim como as linhas e colunas associadas a esse tabuleiro, que me auxiliam a percorrer os N estados anteriores do tabuleiro. Nesta estrutura também guardo a opcao de cada jogador(jogada) assim como a linha e coluna em que foi efetuada essa jogada, para posteriormente a escrever no ficheiro de texto.

4.Opções tomadas em termos de implementação

As minhas funções:

void initRandom(): Esta função inicializa o gerador de numeros aleatorios.

int intUniformRnd(int a, int b): Função que devolve um valor inteiro aleatorio distribuido uniformemente entre [a, b].

char** Cria_matriz_dinamica(char **v, int l, int c): Esta função serve para criar o tabuleiro, que é uma estrutura dinâmica.

char** Aumenta_Linha(char **v, int* l, int c): Função que adiciona uma linha ao fim da estrutura dinâmica.

char **Aumenta_Coluna(char **v, int l, int *c): Função que adiciona uma coluna no fim de cada linha da estrutura dinâmica.

void mostra_tabuleiro(char **jogo, int l, int c): Esta função mostra o tabuleiro na consola.

int caractereValido(char letra): Função que verifica se o caracter numa determinada posição do tabuleiro é ‘G’ ou ‘Y’ ou ‘R’, ou seja os caracteres possiveis para a condição de vitória.

int coordenadaValida(int l, int c, int linhas, int colunas): Função que verifica se a posição que o jogador inseriu pertence ao tabuleiro.

int posicaoValida(char** jogo, int l, int c): Função que verifica se o caracter numa determinada posição do tabuleiro é ‘ ’ ou ‘ G’ ou ‘Y’, ou seja que ainda é permitido jogar na mesma.

int posicaoVazia(char** jogo, int l, int c): Função que verifica se determinada posição do tabuleiro escolhida pelo jogador está vazia.

void escreveJogo(char** jogo, int l, int c): Função que escreve o jogo, ou seja que dependendo das corrdenadas do jogador se nessa posição tiver um ‘ ‘ coloca um ‘G’, se tiver um ‘G’ coloca um ‘Y’ e se tiver um ‘Y’ coloca um ‘R’.

int ganhouLinhas(char **jogo, int linhas, int colunas): Função que verifica se algum jogador venceu por linha.

int ganhouColunas(char **jogo, int linhas, int colunas): Função que verifica se algum jogador venceu por coluna.

int ganhouDiagonalPrincipal(char **jogo, int linhas): Função que verifica se algum jogador venceu pela diagonal principal.

int ganhouDiagonalSecundaria(char **jogo, int linhas): Função que verifica se algum jogador venceu pela diagonal secundária.

int ganhouTudo(char** jogo, int linhas, int colunas): Função que junta o ganhouLinhas, o ganhouColunas, o ganhouDiagonalPrincipal e o ganhouDiagonalSecundaria numa só.

void mostraGanhou(char **jogo, Jogador jogador[2], int linhas, int colunas, int jogadas, int ganhou): Função que mostra na consola qual o jogador que venceu, quando o jogo é efetuado entre duas pessoas.

void mostraGanhouBot(char **jogo, int linhas, int colunas, int jogadas, int ganhou): Função que mostra na consola qual o jogador que venceu, quando o jogo é efetuado entre uma pessoa e um bot.

char turnoJogador(int jogadas): Função que verifica se é a vez do jogador A ou B jogar.

void apagaLista(Lista *tabuleiro): Função que liberta a lista ligada.

void jogar(char **jogo, int linhas, int colunas, int *jogadas, int *jogou, int *l, int *c): Esta função escreve no tabuleiro uma determinda peça consoante as coordenadas inseridas pelo jogador e dependendo do que ja estava nessa posição.

void jogarBot(char **jogo, int linhas, int colunas, int *jogadas, int *jogou, int *l, int *c): Esta função escreve no tabuleiro uma determinda peça consoante as coordenadas inseridas pelo bot e dependendo do que ja estava nessa posição.

void adiciona_pedras(char** jogo, Jogador jogador[2], int linhas, int colunas, int *jogadas, int *jogou, int *l, int *c): Esta função verifica se o jogador que a chamou ainda pode adicionar pedras e caso possa, adicona a na posição escolhida.

void adiciona_pedrasBot(char** jogo, Jogador jogador[2], int linhas, int colunas, int *jogadas, int *jogou, int *l, int *c): Esta função verifica se o bot que a chamou ainda pode adicionar pedras e caso possa, adicona a na posição escolhida aleatoriamente por esse bot.

int Aumentar_Linha_Coluna(Jogador jogador[2], int *jogadas, int *jogou): Função que verifica se o jogador que a chamou ainda pode adicionar linhas ou colunas e caso possa, adicona.

int Aumentar_Linha_ColunaBot(Jogador jogador[2], int *jogadas, int *jogou): Função que verifica se o bot que a chamou ainda pode adicionar linhas ou colunas e caso possa, adicona.

Lista* guardaDados(Lista* aux, char **jogo, int linhas, int colunas, int l, int c, int opcao): Função que guarda na lista ligada os seus respetivos estados referentes ao tabuleiro e

as suas linhas e colunas e em relação a jogada efetuada pelo jogador ou bot numa certa posicao.

void mostraJogadasAnteriores(Lista* tabuleiro, Lista* aux, int n, int jogadas): Função que recebe do jogador o numero de jogadas que deseja rever e caso seja possivel, mostra os respetivos estados do tabuleiro na consola.

void ficheiroTexto(Lista* tabuleiro, Lista* aux, int *turno): Função que escreve num ficheiro de texto a jogada e a linha e coluna em que esta foi efetuada em cada turno.

void ficheirobinario(Lista* tabuleiro, Lista* aux): Função que escreve num ficheiro binário os diferentes estados da lista ligada.

Lista* lebinario(Lista* tabuleiro, Lista* aux): Função que le o ficheiro binário criado anteriormente.

void libertaJogo(char **jogo, int linhas): Função que libertao tabuleiro, ou seja a estrutura dinâmica.

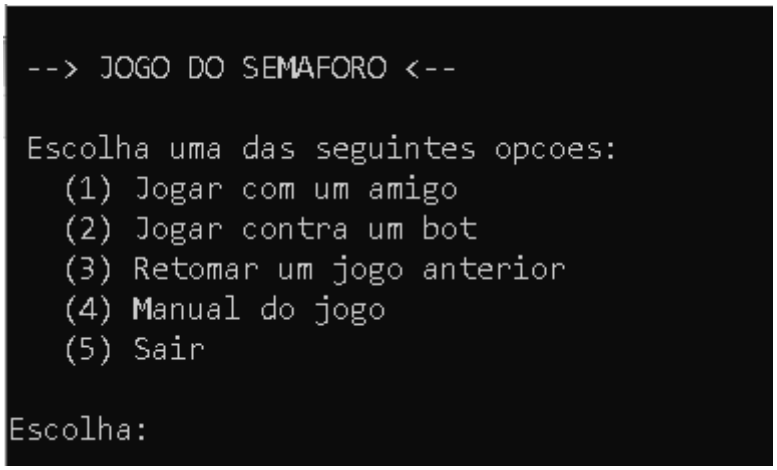
int menu(): Esta funcao disponibiliza um menu, a escolher a opcao e retorna essa opcao.

void jogoNormal(): Esta é uma das funções principais de jogo, que faz o jogo acontecer entre o jogador A e B conforme as suas jogadas.

void jogoBot(): Esta função faz o jogo acontecer entre o jogador e o bot conforme as suas jogadas.

void manual(): Esta função mostra na consola um pequeno manual de funcionamento do jogo.

5.Manual de utilização



Inicialmente é disposto um menu como este ao utilizador, em que dependendo da opção será redirecionado para o seu respetivo destino.

Após o inicio do jogo, este funciona da seguinte maneira:

O jogo realiza-se num tabuleiro, inicialmente vazio.

Em cada jogada, cada jogador realiza uma das seguintes acoes:

- Coloca uma peca verde numa posicao vazio;
- Substitui uma peca verde por uma peca amarela;
- Substitui uma peca amarela por uma peca vermelha.

Cada jogador pode realizar tambem as seguintes jogadas extra:

- Adicionar uma linha ao tabuleiro (maximo de 2 linhas e colunas por jogador);
- Adicionar uma coluna ao tabuleiro (maximo de 2 linhas e colunas por jogador);
- Colocar uma pedra numa posicao vazia (maximo de 1 pedra por jogador).

Cada jogador no inicio da sua jogada podera tambem rever o tabuleiro nas ultimas N jogadas, assim como pedir para pausar o jogo e retoma lo mais tard

6.Conclusão

Concluo esta trabalho com muitos mais conhecimentos sobre a linguagem C do que quando o estava a iniciar, conseguindo agora entender com uma maior amplitude a importância desta linguagem no mundo da programação