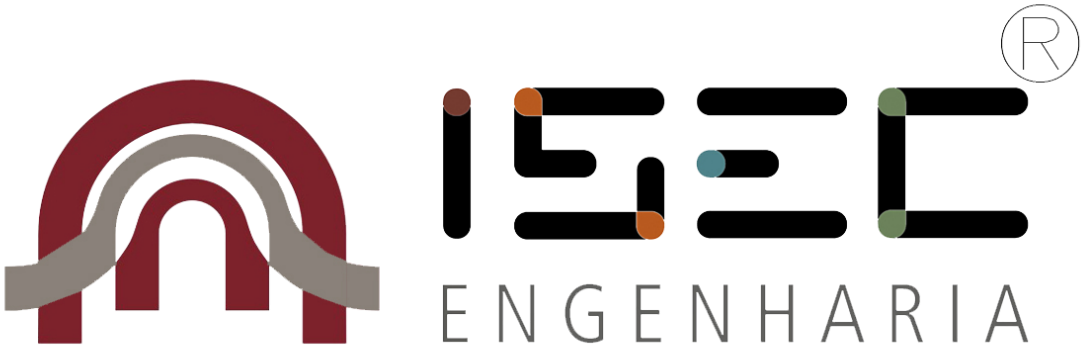


PROGRAMAÇÃO

Jogo do semáforo

Docente: Francisco Pereira

João Miguel Duarte dos Santos

nº 2020136093 LEI

Coimbra, 13 de Junho de 2021

Índice

1. Introdução

Este trabalho surge no âmbito da Unidade Curricular de Programação, no 2º Semestre do ano letivo de 2020/2021. Feito em linguagem C standard, respeitando a norma C99, tem como objetivos principais:

- Desenvolver um jogo para ser disputado entre dois jogadores ou um jogador e o próprio programa;
- Ser de fácil utilização, através da implementação de uma interface simples e amigável, e que esclareça o utilizador em tudo o que pode fazer num determinado momento.

Para alcançar estes objetivos, foram usados conceitos da linguagem C, como estruturas dinâmicas, estruturas ligadas para armazenar informação importante tanto sobre o jogador como as jogadas e estado do tabuleiro, manipulação de ficheiros para guardar e carregar o estado de um jogo e alocamento dinâmico de memória.

2. Estruturas Dinâmicas

As duas estruturas dinâmicas utilizadas foram criadas para guardar informação enquanto o jogo se desenrola:

➤ Foi utilizado um array dinâmico para armazenar o tabuleiro de jogo

➤ Foi utilizado uma lista ligada para guardar todas as informações sobre uma jogada realizada com sucesso

Ambas as estruturas foram alocadas dinamicamente para que seja possível a sua alteração (adicionar elementos ás estruturas) no decorrer do jogo.

Estrutura para guardar todas as informações sobre uma jogada (lista ligada)

A estrutura dinâmica encarregue de guardar toda as informações sobre uma jogada é essencial para que seja possível guardar o estado do jogo ou para que seja possível rever as jogadas passadas.

Esta guarda:

➤ tab ⇢ Ponteiro que aponta para o início do tabuleiro

➤ linhas, colunas ⇢ Número de linhas e colunas do tabuleiro

➤ jogador ⇢ Identificação do jogador que realizou a jogada

➤ jogada ⇢ Identificação da jogada realizada

➤ njogada ⇢ Número da jogada que foi realizada

➤ linhaJogada, colunaJogada ⇢ Posição onde foi colocada a peça

➤ próximo ⇢ Ponteiro que aponta para o próximo elemento da lista

```c
typedef struct dados tabu, *pnttabuleiro;
struct dados
{
    char **tab;             // ponteiro para o inicio do tabuleiro
    int linhas;             // Número de linhas
    int colunas;            // Número de colunas
    char jogador;           // Caracter do jogador
    int jogada;             // Jogada realizada
    int linhaJogada, colunaJogada;  // Linha e coluna escolhidas na jogada
    int njogada;            // Número da jogada
    pnttabuleiro proximo;       // ponteiro para o próximo elemento
};
```

Array dinâmico para armazenar o tabuleiro de jogo

Todo o tabuleiro é guardado e armazenado de forma dinâmica através da alocação dinâmica de memória (através da função malloc).

É usada a alocação dinâmica de memória para que este array bidimensional possa ser acessado e alterado em qualquer altura do jogo usando a função expandirLin (para aumentar o número de linhas) ou expandirCol (para aumentar o número de colunas) com recurso á função realloc.

A função responsável por este procedimento é a gerarTab:

```
int i,j;            // Variáveis auxiliares para percorrer o tabuleiro
char **tabuleiro;   // Variável para armazenar o tabuleiro

tabuleiro=(char**)malloc(sizeof(char*)*linhas); // Alocação de memória das linhas (cada uma com tamanho para um char*) do tabuleiro

if(tabuleiro==NULL){    // Caso não haja memória disponivel
    printf("Erro na alocacao de memoria!"); // Erro
    exit(0);
}

for(i=0;i<linhas;i++){  // Para cada linha do tabuleiro
    tabuleiro[i]=(char *)malloc(sizeof(char)*colunas);  // Alocação de memória para colunas(cada uma com tamanho para um char)

    if(tabuleiro[i]==NULL){ // Caso não haja memória disponivel
        printf("Erro na alocacao de memoria!"); // Erro
        exit(0);
    }
}

for(i=0;i<linhas;i++){
    for(j=0;j<colunas;j++){     // Percorrer todas as linhas e colunas do tabuleiro
        tabuleiro[i][j]='-';    // Colocar o caracter '-' em todos os espaços do tabuleiro (torná-lo vazio)
    }
}
return tabuleiro;   // Retorna ponteiro para o início do tabuleiro de jogo
```

Esta função deixa todos os espaços vazios com o caracter '-'.

No fim do jogo é essencial que a memória reservada para armazenar o tabule iro seja libertada, e daí o uso da função free.

```
free(tab);  // Libertar memória alocada para o tabuleiro
```

3. Estruturas de Dados

Estrutura para guardar informações sobre um jogador

Apesar de ser uma estrutura muito simples, é importante para que seja possível manter uma "noção" de quem foi o último jogador a realizar uma jogada e saber quem será o próximo jogador a jogar.

Esta guarda:

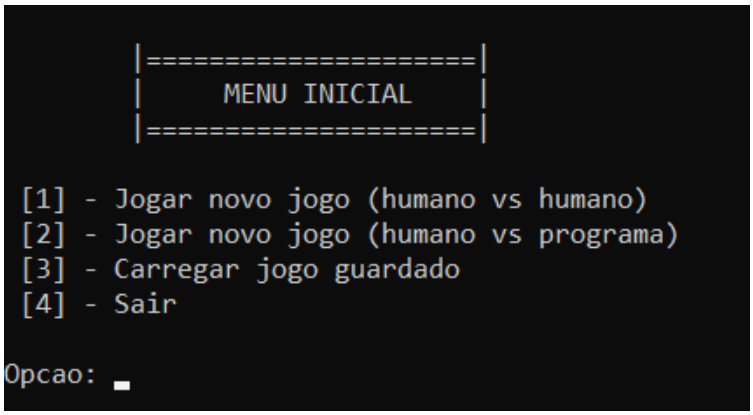
➤ jogador ⇢ Caracter para identificar o jogador

```
typedef struct JOGADOR jogador;
struct JOGADOR{
    char jogador;
};
```

4.Implementações

Menu inicial

Logo após iniciar o programa, é mostrado o menu inicial (composto por um swtich-case) com várias opções (novo jogo contra humano, novo jogo contra o computador, carregar um jogo anteriormente guardado e a opção de sair e fechar o programa).



Ao digitar 1 e pressionar enter, selecionamos a opção de jogar um novo jogo contra um humano e é chamada a função jogar que é responsável pelo desenrolar do jogo (contra humano).

A tecla 2 chama a função jogarComputador que é responsável pelo desenrolar do jogo (conta o próprio programa). Esta função inclui a realização de jogadas automáticas realizadas pelo próprio programa.

A terceira opção chama a função retomarJogo que procura algum ficheiro guardado anteriormente (com o nome jogo.bin) e o abre para que seja possível continuar a jogar a partir desse registo guardado.

A quarta opção simplesmente fecha o programa.

Gerar o tabuleiro

A função responsável por gerar o tabuleiro é a gerarTab que recebe como parâmetros o número de linhas e colunas que terá o tabuleiro depois de ser gerado.

É criado um ponteiro para armazenar o tabuleiro e depois temos a alocação de memória para as linhas e colunas.

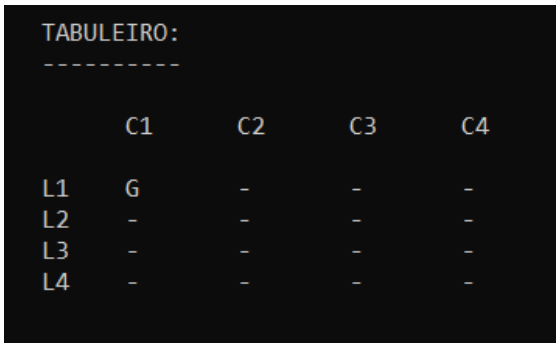
Esta função preenche ainda os espaços vazios com o caracter '-' e retorna o ponteiro que aponta para o início do tabuleiro.

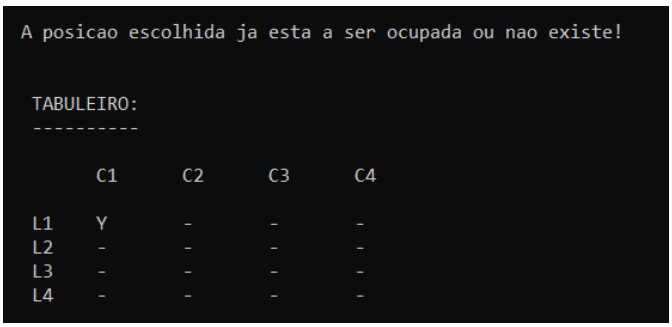
```
char **gerarTab(int linhas,int colunas)
{
    int i,j;            // Variáveis auxiliares para percorrer o tabuleiro
    char **tabuleiro;   // Variável para armazenar o tabuleiro

    tabuleiro=(char**)malloc(sizeof(char*)*linhas); // Alocação de memória das linhas (cada uma com tamanho para um char*) do tabuleiro

    if(tabuleiro==NULL){    // Caso não haja memória disponivel
        printf("Erro na alocacao de memoria!"); // Erro
        exit(0);
    }

    for(i=0;i<linhas;i++){  // Para cada linha do tabuleiro
        tabuleiro[i]=(char *)malloc(sizeof(char)*colunas);  // Alocação de memória para colunas(cada uma com tamanho para um char)

        if(tabuleiro[i]==NULL){ // Caso não haja memória disponivel
            printf("Erro na alocacao de memoria!"); // Erro
            exit(0);
        }
    }

    for(i=0;i<linhas;i++){
        for(j=0;j<colunas;j++){     // Percorrer todas as linhas e colunas do tabuleiro
            tabuleiro[i][j]='-';    // Colocar o caracter '-' em todos os espaços do tabuleiro (torná-lo vazio)
        }
    }
    return tabuleiro;   // Retorna ponteiro para o início do tabuleiro de jogo
}
```

Menu de jogadas

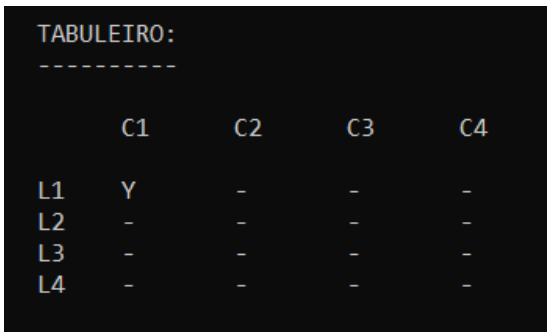
Ao selecionar umas das opções do menu inicial com o objetivo de jogar, aparece-nos o tabuleiro gerado pela função gerarTab, a informação do jogador que realizará a próxima jogada e o menu de jogadas (composto por um switch--case) com opções das várias jogadas disponíveis.

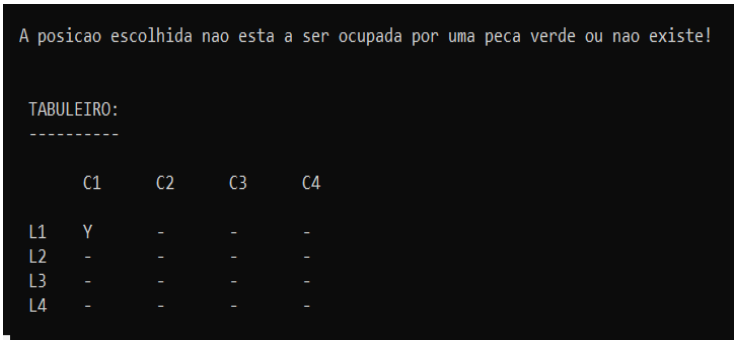
A opção 1 pede o número da linha e coluna onde o utilizador quer colocar a peça verde. Depois verifica se esses número são possíveis (se a linha e coluna escolhidas fazem parte do tabuleiro). Se não forem possíveis, apresenta uma mensagem de erro e volta a perguntar a linha e coluna. Se for possível então se a posição estiver vazia, coloca a peça verde, mas se não, apresenta uma mensagem de erro e mostra novamente o menu de jogadas.





A opção 2 pede o número da linha e coluna onde o utilizador quer colocar a peça verde. Depois verifica se esses número são possíveis (se a linha e coluna escolhidas fazem parte do tabuleiro). Se não forem possíveis, apresenta uma mensagem de erro e volta a perguntar a linha e coluna. Se for possível então se a posição estiver ocupada por uma peça verde, coloca a peça amarela, mas se não, apresenta uma mensagem de erro e mostra novamente o menu de jogadas.





A opção 3 pede o número da linha e coluna onde o utilizador quer colocar a peça verde. Depois verifica se esses número são possíveis (se a linha e coluna escolhidas fazem parte do tabuleiro). Se não forem possíveis, apresenta uma mensagem de erro e volta a perguntar a linha e coluna. Se for possível então se a posição estiver ocupada por uma peça amarela, coloca a peça vermelha, mas se não, apresenta uma mensagem de erro e mostra novamente o menu de jogadas.

```
TABULEIRO:
----------

      C1      C2      C3      C4

L1    Y       -       -       -
L2    -       -       -       -
L3    -       -       -       -
L4    -       -       -       -


           || Jogador A ||

      |=========================|
      |      MENU DE JOGADAS    |
      |=========================|

[1] - Colocar peca verde numa celula vazia do tabuleiro
[2] - Trocar uma peca verde do tabuleiro por uma amarela
[3] - Trocar uma peca amarela do tabuleiro por uma vermelha
[4] - Colocar uma pedra numa celula vazia do tabuleiro
[5] - Adicionar uma linha vazia ao final do tabuleiro
[6] - Adicionar uma coluna vazia ao final do tabuleiro
[7] - Rever jogadas (maximo de 3 jogadas anteriores)
[8] - Guardar jogo e sair para o menu inicial

Opcao: 3

Linha: 1
Coluna: 1
```

```
TABULEIRO:
----------

      C1      C2      C3      C4

L1    R       -       -       -
L2    -       -       -       -
L3    -       -       -       -
L4    -       -       -       -
```

```
A posicao escolhida nao esta a ser ocupada por uma peca amarela ou nao existe!


TABULEIRO:
----------

     C1    C2    C3    C4

L1   R     -     -     -
L2   -     -     -     -
L3   -     -     -     -
L4   -     -     -     -
```

A opção 4 pede o número da linha e coluna onde o utilizador quer colocar a peça verde. Depois verifica se esses número são possíveis (se a linha e coluna escolhidas fazem parte do tabuleiro). Se não forem possíveis, apresenta uma mensagem de erro e volta a perguntar a linha e coluna. Se for possível então se a posição estiver vazia, coloca uma pedra, impossibilitando a vitória através dessa linha, coluna ou diagonal se for o caso. Se não, apresenta uma mensagem de erro e mostra novamente o menu de jogadas.

Caso o jogador tenha usado esta jogada 1 vez, da próxima que escolher esta opção aparece uma mensagem de erro.

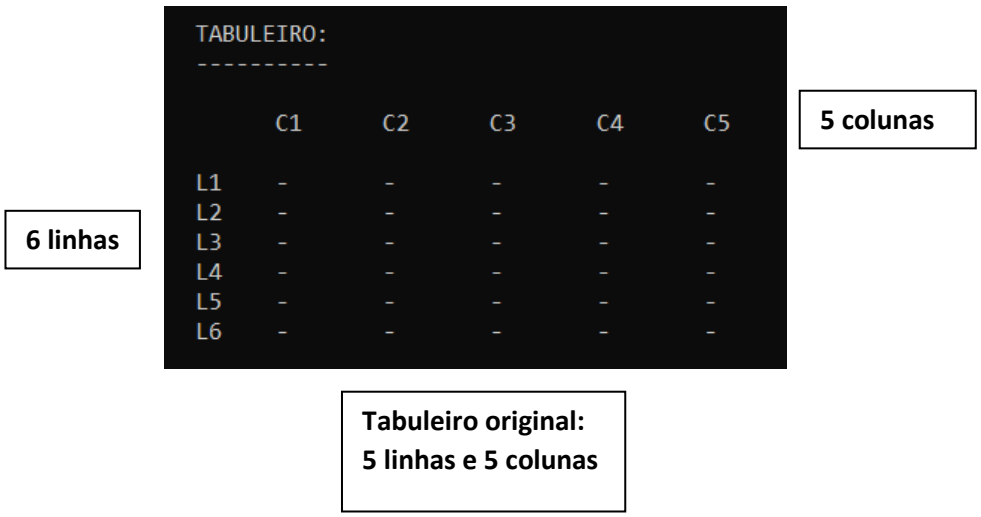
```
TABULEIRO:
----------

      C1      C2      C3      C4

L1    -       -       -       -
L2    -       -       -       -
L3    -       -       -       -
L4    -       -       -       -


           || Jogador A ||

       |=========================|
       |      MENU DE JOGADAS    |
       |=========================|

[1] - Colocar peca verde numa celula vazia do tabuleiro
[2] - Trocar uma peca verde do tabuleiro por uma amarela
[3] - Trocar uma peca amarela do tabuleiro por uma vermelha
[4] - Colocar uma pedra numa celula vazia do tabuleiro
[5] - Adicionar uma linha vazia ao final do tabuleiro
[6] - Adicionar uma coluna vazia ao final do tabuleiro
[7] - Rever jogadas (maximo de 3 jogadas anteriores)
[8] - Guardar jogo e sair para o menu inicial

Opcao: 4

Linha: 1
Coluna: 1
```

```
TABULEIRO:
----------

      C1      C2      C3      C4

L1    x       -       -       -
L2    -       -       -       -
L3    -       -       -       -
L4    -       -       -       -
```

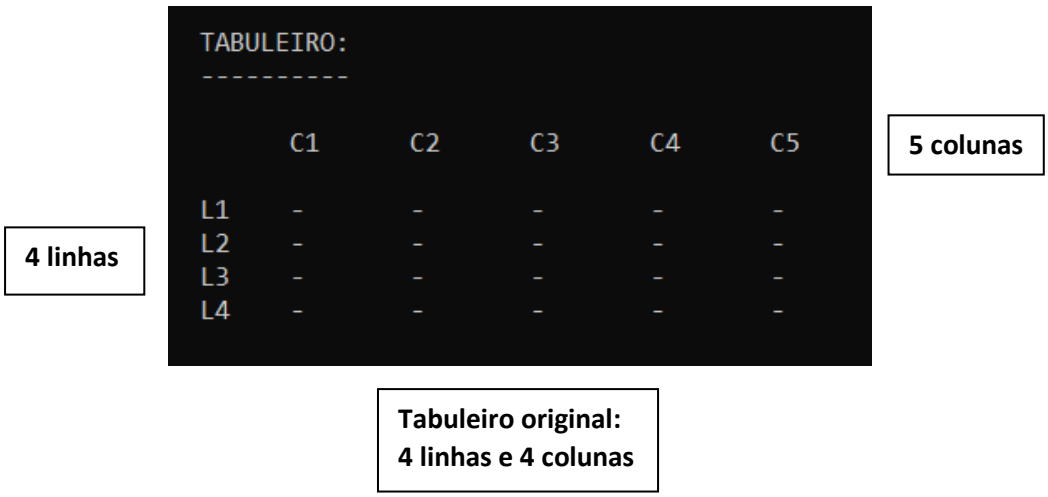
```
A posicao escolhida ja esta a ser ocupada ou nao existe!


TABULEIRO:
----------

     C1      C2      C3      C4

L1   x       -       -       -
L2   -       G       G       -
L3   -       -       -       -
L4   -       -       -       -
```

```
Esta jogada so pode ser escolhida uma vez por jogador!
Escolha outra opcao.


TABULEIRO:
----------

      C1      C2      C3      C4

L1    x       -       -       -
L2    -       -       G       -
L3    -       -       -       -
L4    -       -       -       -
```

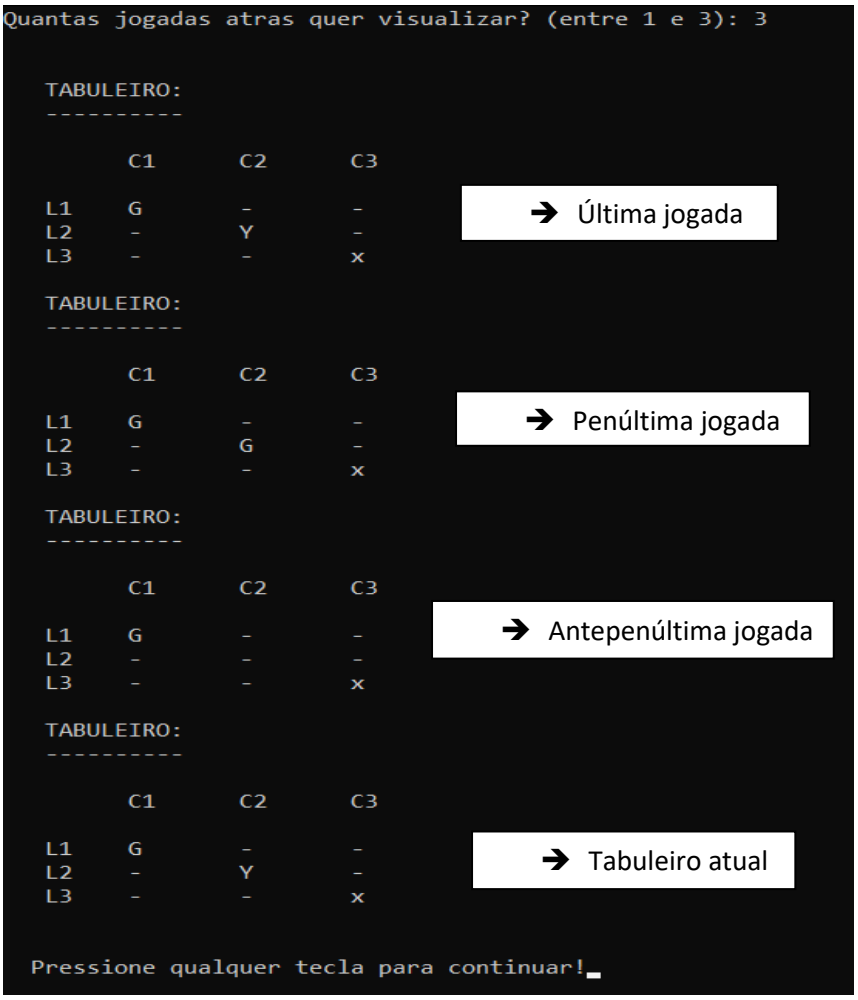
A opção 5 chama a função expandirLin responsável pelo aumento do tabuleiro (adiciona uma linha) através da função realloc que realoca um novo espaço de memória para a nova linha e as novas colunas para essa linha. Retorna um ponteiro para o início do tabuleiro já atualizado.



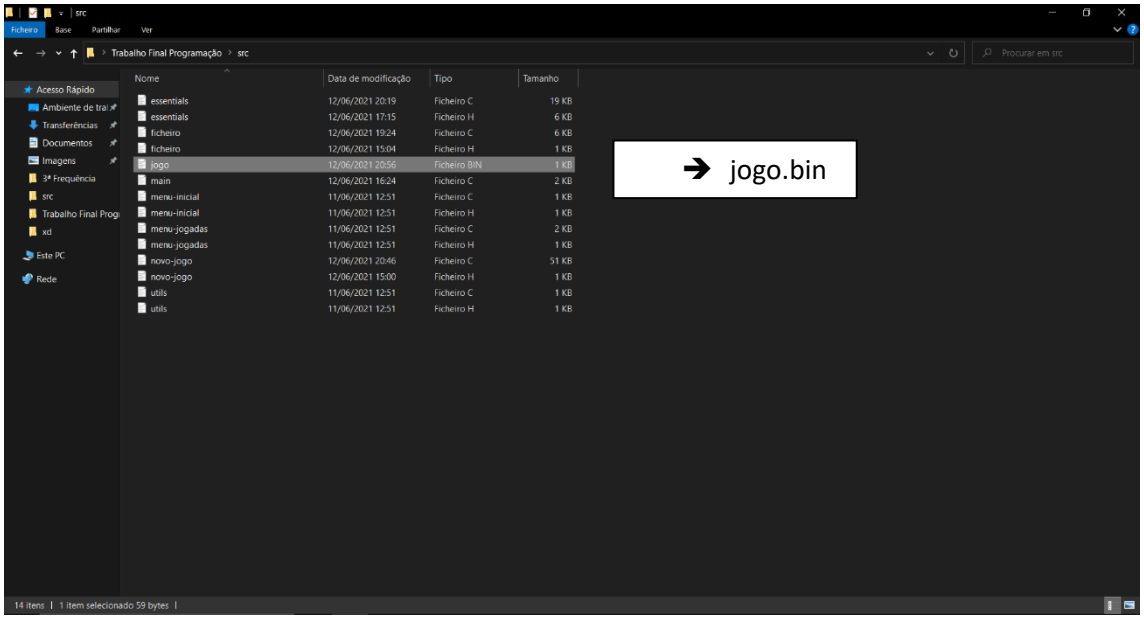
A opção 6 chama a função expandirCol responsável pelo aumento do tabuleiro (adiciona uma linha) através da função realloc que realoca um novo espaço de memória para a nova coluna e as novas linhas para essa coluna. Retorna um ponteiro para o início do tabuleiro já atualizado.



A opção 7 pede o número de jogadas anteriores que o utilizador deseja rever. Depois chama a função inverteLista que inverte a lista ligada que contém toda a informação de todas as jogadas já realizadas para que fique por ordem crescente em relação ao decorrer do jogo, ou seja, a primeira jogada é o primeiro elemento da lista ligada. Retorna o número de elementos presentes na lista. A seguir mostra na tela as jogadas anteriores que o utilizador escolheu rever.



A opção 8 chama a função guardaJogo responsável por criar um ficheiro com o nome “ficheiro.bin” que guarda o estado do tabuleiro naquele ponto, o jogador que realizou a última jogada e retorna 1 se tiver sucesso. Depois disso, é chamada a função liberta_memlista responsável por libertar a memória alocada para a lista ligada.



5.Conclusão

Este trabalho prático deu-me algum prazer a realizar, pois, para além de gostar bastante de programação era sobre criar um jogo, o que deu mais vontade ainda de trabalhar.

Permitiu-me aplicar os conteúdos que foram dados nas aulas teóricas e práticas e também a “obrigação” de pesquisar e me informar sobre assuntos e outros conteúdos úteis através da internet.