Programação

Trabalho Prático – Jogo do semáforo

LEI - 2020/2021

João Pedro Caldas Cerqueira

Nº: 2020141650

Índice

Interface 3
Opções de jogo: 4
1- Colocar uma peça 4
2- Colocar uma pedra 4
3- Adicionar linha 4
4- Adicionar coluna 4
5- Ver jogadas anteriores 5
6- Termina jogo 5
Estruturas 5
Estruturas de dados 5
- Struct jog: 5
- Struct ret: 6

..... 6
- Struct joga: 6

Estruturas dinâmicas 7
- Array dinâmico: 7
- Lista ligada 8

Interface

Ao iniciar o jogador tem a opção de ver as regras do jogo clicando na tecla 'R'.

```
Se for a sua primeira vez a jogar prima a tecla R para verificar as regras, caso contrario prima enter.

R

O jogo do semaforo desenrola-se num tabuleiro dividido em celulas. No inicio, o tabuleiro esta vazio. A
lternadamente os jogadores vAo colocando peþas de cor Verde (G), Amarela (Y) ou Vermelha (R). Ganha o j
ogador que coloque uma peþa que permita formar uma linha,coluna ou diagonal completa com pecas da mesma
 cor. As jogadas validas relativas acolocacao de pecas sao as seguintes:
        1. Colocar uma peca Verde numa celula vazia;
        2. Trocar uma peca Verde que esteja colocada no tabuleiro por uma peca Amarela;
        3. Trocar uma peca Amarela que esteja colocada no tabuleiro por uma peca Vermelha;

Existem duas jogadas adicionais que podem ser efetuadas pelos jogadores:        4. Colocar uma pedra nu
ma celula vazia. Cada jogador pode colocar, no maximo, umapedra por jogo. A colocacao de uma pedra invi
abiliza que o jogo possa terminar porpreenchimento da linha e coluna afetadas (e, eventualmente tambÚm
da diagonal oudiagonais).  5. Adicionar uma linha ou uma coluna ao final do tabuleiro. Esta jogada adic
iona linhasou colunas completas e vazias ao tabuleiro de jogo. Cada jogador pode efetuar estajogada, no
 maximo, duas vezes por jogo.

Em cada iteracao, um jogador escolhe uma destas jogadas para atualizar o tabuleiro. Asjogadas 4 e 5 est
ao sujeitas as restricoes indicadas na sua definicao. O numero de pecas decada cor e ilimitado.
```

De seguida o jogador pode iniciar um novo jogo ou continuar o jogo anterior e caso escolha iniciar novo jogo tem a opção de selecionar o modo (jogador 1 VS jogador 2 ou jogador 1 VS computador).

```
        1.Novo Jogo
        2.Continuar jogo
1
Selecione o modo de jogo:
        1. Jogador A vs Jogador B
        2. Jogador A vs Computador
```

Ao iniciar o jogo é exibido o tabuleiro (que pode ser 3x3, 4x4, 5x5), o número de linhas/colunas que cada jogador pode acrescentar ao tabuleiro e ainda o número de pedras disponíveis.

```
-------------------------------------
jogador A:
Linhas ou Colunas extra disponiveis: 2
Pedras disponiveis: 1

jogador B:
Linhas ou Colunas extra disponiveis: 2
Pedras disponiveis: 1
-------------------------------------


  Tabuleiro 4x4

     C0    C1    C2    C3
L0 [ _ ] [ _ ] [ _ ] [ _ ]
L1 [ _ ] [ _ ] [ _ ] [ _ ]
L2 [ _ ] [ _ ] [ _ ] [ _ ]
L3 [ _ ] [ _ ] [ _ ] [ _ ]
```

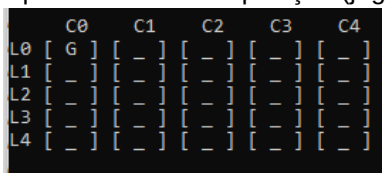
Opções de jogo:

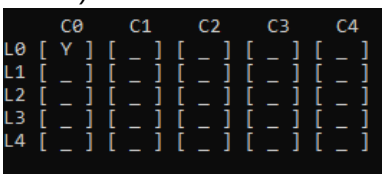
Cada jogador tem ao seu dispor 6 opções sendo elas:

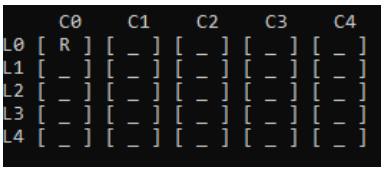
1- Colocar uma peça

O jogador escolhe a linha e a coluna onde deseja colocar a peça.

- Caso o lugar esteja vazio a peça colocada é verde (G);
- Caso o lugar esteja ocupado com uma peça verde é substituída por uma peça amarela (Y);
- Caso o lugar esteja ocupado com uma peça amarela é substituída por uma peça vermelha (R);
- Caso o lugar esteja ocupado com uma peça vermelha o jogador terá que escolher outra posição (jogada inválida).



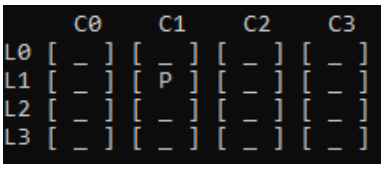




2- Colocar uma pedra

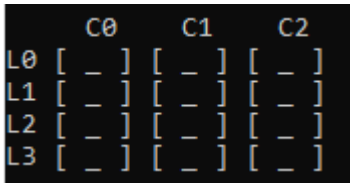
O jogador escolhe a linha e a coluna onde deseja colocar a pedra, caso ainda tenha a pedra disponível.

- Caso o lugar esteja vazio é colocada a pedra (P);
- Caso o lugar esteja ocupado por uma peça terá de escolher outra posição (jogada inválida).



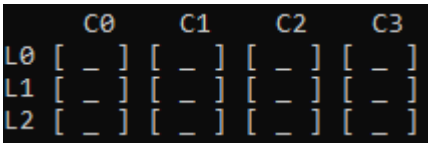
3- Adicionar linha

- Caso o jogador ainda tenha linhas extra disponível, é adicionada uma linha ao tabuleiro.
- Caso contrário o jogador tem a possibilidade de escolher outra opção.



4- Adicionar coluna

- Caso o jogador ainda tenha colunas extra disponível, é adiciona uma coluna ao tabuleiro.
- Caso contrário o jogador tem a possibilidade de escolher outra opção.



5- Ver jogadas anteriores

O jogador tem a opção de visualizar o número de jogadas anteriores que escolher.

```
numero de jogadas que quer ver: 2
```

```
jogada 4
  Tabuleiro 3x3

     C0    C1    C2
L0 [ Y ] [ G ] [ _ ]
L1 [ G ] [ G ] [ _ ]
L2 [ _ ] [ _ ] [ _ ]
```

```
jogada 5
  Tabuleiro 3x3

     C0    C1    C2
L0 [ Y ] [ G ] [ G ]
L1 [ G ] [ G ] [ _ ]
L2 [ _ ] [ _ ] [ _ ]
```

6- Termina jogo

Se o jogador quiser interromper o jogo, pode terminá-lo. As jogadas realizadas são guardadas num ficheiro binário (jogo.bin) para que posteriormente o possa retomar.

Quando um jogador vencer o jogo, vai ser pedido para atribuir um nome a um ficheiro.txt (onde vai conter toda a informação das jogadas realizadas).

Estruturas

Estruturas de dados

- Struct jog:

  Esta estrutura guarda o nome de um jogador, o número de colunas/linhas que ainda tem disponível para usar e a pedra.
  A estrutura é bastante útil pois vai guardando informação necessária de cada jogador ajuda o utilizador a saber por exemplo se ainda pode colocar a pedra ou não.

```c
typedef struct jog jogador;

struct jog{
    char nome[15];
    int Col_linExtra;
    int pedra;
};
```

- Struct ret:

  Esta estrutura têm o objetivo guardar a informação do jogo para que o utilizador possa terminar o jogo a meio e posteriormente retomar.
  Os dados da estrutura são mandados para um ficheiro binário que quando o jogador quiser retomar o jogo são recuperados e o jogo recomeço no mesmo ponto que terminou.
  A estrutura guarda os 2 jogadores, o contador para que quando o jogo retome se saiba em que jogador ficou, também guarda o tabuleiro da última jogada e as suas respetivas linhas e colunas.

```
typedef struct ret retoma;

struct ret{
    jogador a;
    jogador b;
    int contador;
    int lin;
    int col;
    char **tabuleiro;
};
```

- Struct joga:

  Esta estrutura é bastante importante pois tem como função guardar a jogada feito por um jogador e posteriormente será usada para construir a lista ligada com todas as jogadas efetuadas no jogo.

  A estrutura guarda o contador, o tabuleiro e respetivamente o número de linhas e colunas e por fim guarda também um ponteiro que vai servir para ligar as jogadas todas.

```
typedef struct joga jogadas, *pjogadas;

struct joga{
   int contador;
   int lin;
   int col;
   char**tab;
   pjogadas prox;
};
```

Estruturas dinâmicas

- Array dinâmico:

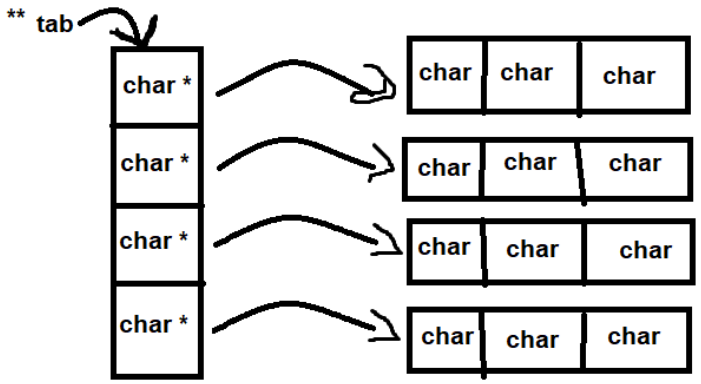
Tabuleiro:

```
char **tabuleiro;

tabuleiro=(char**) malloc(sizeof(char*)*lin);
for(i=0;i<lin;i++){
    tabuleiro[i]=malloc(sizeof(char)*col);
}
```

Optei por criar o tabuleiro a partir de um Array dinâmico visto que estamos habituados a trabalhar com arrays e pareceu me mais fácil e também porque depois se quiser mexer nele é só fazer como se faz para os arrays normais.

- Em primeiro lugar inicializei o tabuleiro como um ponteiro para ponteiro do tipo char (char **tabuleiro).
- De seguida aloquei espaço para as linhas colocando o **char a apontar para um array de *char que vão ser as linhas.
- De seguida coloquei os *char a apontar uma um array de char que vão ser as colunas.



Para adicionar linhas preciso de realocar o tabuleiro (+ 1 linha) então aumento uma linha, mas depois também tenho que alocar espaço para as colunas dessa linha.

```
aux =(char**) realloc(tab,sizeof(char*)*(*lin+1));
aux[*lin] = malloc(sizeof(char)*(*col));

if(aux==NULL){
    printf("erro na realocacao");
    return tab;
}

for(i=0;i<*col;i++){
    aux[*lin][i]='_';
}

tab=aux;
(*lin)++;
```

Para adicionar linhas preciso de realocar o tabuleiro (+ 1 coluna). Neste caso basta realocar espaço para mais uma coluna em cada linha.

```
for(i=0;i<(*lin);i++){
    tab[i]= realloc(tab[i],sizeof(char)*(*col+1));
    tab[i][*col]='_';
}
```

- Lista ligada
  - Para criar uma lista ligada primeiro precisamos de uma estrutura de dados que neste caso é a struct joga que falei anteriormente.
  - Depois precisamos de inicializar a lista a NULL.
  - Para adicionar uma nova jogada na lista (adicionar no final) precisamos:
    1. **Alocar espaço para a jogada.**

```
//alocar espaco para nova jogada
novo=malloc(sizeof(jogadas));
if(novo==NULL)
    return p;
```

    2. **Preencher a jogada.**

```
//prencher a jogada
novo->contador=contador;

novo->lin=lin;
novo->col=col;

novo->tab = malloc(sizeof(char*)*lin);
            for(int i = 0; i < lin; i++)
                novo->tab[i] = malloc(sizeof(char)*col);

for(i=0;i<lin; i++){
    for(int j=0;j<col;j++){
        novo->tab[i][j]=tab[i][j];
    }
}
novo->prox=NULL;
////////////////////////////////
```

    3. **Colocar jogada no final da lista.**

```
//colocar jogada no final
if(p==NULL)
    p=novo;
else{
    aux=p;
    while(aux->prox!=NULL)
        aux=aux->prox;

    aux->prox=novo;
    novo->prox=NULL;
}
return p;
```