



## Programação

Trabalho Prático 2021/2022

Docente:

Francisco Pereira

Alunos:

Paulo Henrique Figueira Pestana de Gouveia nº 2020121705 – LEI

Coimbra, 17 de Junho de 2022

# Índice

<b>1. Introdução</b>	<b>2</b>
<b>2. Desenvolvimento</b>	<b>2</b>
2.1 Estrutura	2
2.1.1 Coordenadas Locais Convertidas	2
2.2 Estrutura Dinâmica	2
2.2.1 Input dos Jogadores	2
2.3 Ficheiros do Programa	3
2.3.1 main.c	3
2.3.2 startGame.c	3
2.3.2 criaRegras.c	4
2.3.3 criaTabuleiro.c	4
2.3.4 playingGame.c	5
2.3.5 tratar_input.c	6
2.3.5 winner.c	7
2.3.6 listas_ligadas.c	7
2.3.7 ficheiros.c	8
2.3.8 pausar.c	8
2.3.8 recuperar_jogo.c	8
3.1 Menu	9
3.2 Jogo	9
3.2.1 Fazer uma Jogada	10
3.2.2 Rever uma Jogada ou mais Jogadas	11
3.2.3 Pausar o jogo	11
3.2.4 Vencedor	11
3.3 Recuperação do Jogo Anterior	13

# 1. Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Programação e consiste na criação do “Ultimate Tic-Tac-Toe” feito na linguagem C99.

## 2. Desenvolvimento

### 2.1 Estrutura

#### 2.1.1 Coordenadas Locais Convertidas

Esta estrutura é responsável por armazenar as coordenadas convertidas depois da jogada de um jogador. Isto é, para facilitar o jogador a pensar em cada tabuleiro localmente, as jogadas possíveis são de 1 a 3 para linha e coluna de cada mini tabuleiro, mas o tabuleiro é bidimensional é de [0-8][0-8]. São efetuados cálculos para converter essas coordenadas e estas são as variáveis responsáveis por as guardar.

```
typedef struct todascoordenadas
{
    int n_miniConverted; //minitabuleiro convertido para 0 a 8
    int x_mini_tabuleiro; //pegar no x da jogada e convertê-la para tabuleiro global
    int y_mini_tabuleiro; //pegar no y da jogada e convertê-la para tabuleiro global
    char winnerArray[10]; //array que guarda os minitabuleiros ganhos ou empatados
} coordenadas,*pcoordenadas;
```

Figura 1

### 2.2 Estrutura Dinâmica

#### 2.2.1 Input dos Jogadores

A estrutura dinâmica é a que armazena cada jogada do jogador válida, incluído o seu turno. É conseguido pelo método de lista ligada em que o “pjogadas prox” é usado como ponteiro que aponta para si mesmo para guardar em seguida a sua próxima jogada.

Após a realização de uma jogada é então armazenada a informação que achei necessária para a lista, sendo de seguida criado um novo espaço para armazenar a próxima jogada que aponta para NULL.

```

typedef struct dados_jogada jogadas,*pjogadas;

struct dados_jogada
{
    int turnos;
    int mini_tabuleiro;
    char *input_jogadas;
    int x;
    int y;

    pjogadas prox;
};

```

Figura 2

## 2.3 Ficheiros do Programa

O trabalho está dividido em vários ficheiros usando um **header.h** para os conectar. Em cada ponto é brevemente explicado as funções presentes em cada **ficheiro.c**, que funções chamam e porquê, e a maneira que o jogo decorre.

### 2.3.1 main.c

No **main.c** inicia o random("initRandom();") e chama a função **void start\_Game()** que irá dar começo ao jogo.

### 2.3.2 startGame.c

**void start\_Game()** inicializa a struct “**dados**” em forma da struct “**jogadas**”, inicia a struct “**pdados**” que armazena o endereço de “**dados**” em forma de “**pjogadas**” que é um ponteiro para a struct “**jogadas**” será onde ficaram armazenados o input de cada jogada. Inicia também a “**lista**” a apontar para NULL em forma de “**pjogadas**” que será a struct responsável pela nossa lista ligada.

É feita a display do nosso tabuleiro de regras pela função **void regrasDoJogo()** e também o de jogo pela função **char\*\* criaMat(int nLin, int nCol)**.

Após isto, é perguntado ao utilizador o modo de jogo(Jogador vs Jogador || Jogador vs Computador) em cada uma é mostrado o tabuleiro inicial através da função **void mostraMat(char \*\*p, int nLin, int nCol)**.

São direcionados para a função **void playing\_game(char \*\*tabuleiro, pjogadas pdados, pjogadas lista, int robo)** que dá continuidade a jogar o jogo com um “**int robo**” para identificar o modo de jogo.

Nesta função é gerado um random do primeiro mini-tabuleiro a ser jogado.

### 2.3.2 criaRegras.c

Contém as funções responsáveis por dar setup ao tabuleiro de regras, **void regrasDoJogo()** é função chamada para dar início às outras funções do ficheiro.

**char\*\* criaMatRegras(int nLin,int nCol)** cria o tabuleiro de regras, é declarado de um duplo ponteiro “**char\*\* p**” responsável pelo tabuleiro bidimensional regras, é feito um “**malloc(sizeof(char)\* nLin)**” para armazenar ponteiros de “**char**” num array com o mesmo tamanho das linhas do tabuleiro. Com um ciclo “**for**” percorre cada índice desse mesmo array e efetua um “**malloc(sizeof(char) \* nCol)**” ao chegar ao final é criado o array bidimensional com tamanho correto e esse mesmo é retornado.

**void setupTabuleiroRegras(char\*\* pregras)** enumera os mini-tabuleiros através da função **void setPos(char \*\*p, int x, int y, char c)** que mete os números nas posições dadas.

O print do tabuleiro é efetuado pela função **mostraMat(char \*\*p, int nLin, int nCol)**.

No final como não será mais utilizado é chamada a função **void libertaMat(char\*\* p, int nLin)** para fazer o free do array bidimensional regras.

### 2.3.3 criaTabuleiro.c

As funções neste ficheiros são a responsáveis pela criação, impressão e o free do nosso tabuleiro de jogo.

**char\*\* criaMat(int nLin,int nCol)** cria o tabuleiro de jogo, é declarado de um duplo ponteiro “**char\*\* p**” responsável pelo tabuleiro bidimensional jogo, é feito um “**malloc(sizeof(char)\* nLin)**” para armazenar ponteiros de “**char**” num array com o mesmo tamanho das linhas do tabuleiro. Com um ciclo “**for**” percorre cada índice desse mesmo array e efetua um “**malloc(sizeof(char) \* nCol)**” ao chegar ao final é criado o array bidimensional com tamanho correto e esse mesmo é retornado.

**void mostraMat(char\*\* p, int nLin, int nCol)** percorre o array bidimensional e dá print da forma configurado pelo programador através de um ciclo “**for**”.

**void libertaMat(char\*\* p, int nLin)** liberta memória de um array através do “**free**”, que neste caso para libertar arrays alocados em memória pelo programa.

### 2.3.4 playingGame.c

É aqui onde começa os ciclos das jogadas, e a verificação do ficheiro binário de jogos anteriores.

É criada a struct “coordenadas” em forma da struct “coordenadas” e o “pcoordenadas” em forma de “pcoordenadas” que recebe como endereço a struct “coordenadas”.

O winnerArray é armazenado na “pcoordenadas” que vai ser o array responsável por manter os mini-tabuleiros completos na posição correta.

De seguida é feita a verificação se existe um ficheiro binário de um jogo anterior através do **bool verificarbin()**. Se não existe, o jogo decorre normalmente, caso contrário, é chamda função **bool querContinuar()** para saber se quer continuar o jogo anterior.

Caso o utilizador queira continuar é feita a reconstrução da lista e das jogadas efetuados pelo jogador anterior através da função **pjogadas recuperarJogo(pjogadas lista, pjogadas pdados,char \*\*tabuleiro, pcoordenadas pcoordenadas)**.

Em ambos os casos o jogo entra no mesmo ciclo “while” a diferença será que a jogada começaria no ponto em que ficou no jogo anterior e com a tabuleiro já preenchido.

O ciclo “while” verifica a char retornada pela função **char checkWinner(pcoordenadas pcoordenadas)** é que caso haja um winner manda o char do jogador que ganhou, em caso de empate manda “!” e saí do “while”.

Ao sair do “while” usa uma condição “if” para saber que tipo de vitória foi, ou se foi empate, ao ser vitória usa função **checkTurnos(pjogadas pdados)** para saber o jogador que venceu.

De seguida chama a função **bool ficheiro\_texto(pjogadas lista)** que gera o ficheiro de texto final do jogo e é feita a libertação de memória ocupado pelo tabuleiro de jogo através do **void libertaMat(char\*\* p, int nLin)** e libertação da memória ocupada pela lista ligada **void elimina\_lista(pjogadas lista)**.

Dentro do “while” é feita a verificação do jogador a jogar através da função **bool checkTurnos(pjogadas pdados)** que faz a comparação do resto da divisão inteira por 2 dos turnos e devolve true ou false para informar que jogador tem o turno. Sabendo o jogador a efetuar a jogada chamamos a função **pjogadas antesdeJogada(char\*\* tabuleiro,pjogadas pdados,pjogadas lista,int robo,pcoordenadas pcoordenadas)** que pergunta ao utilizador que tipo de jogada quer.

A função **pjogadas antesdeJogada(char\*\* tabuleiro,pjogadas pdados,pjogadas lista,int robo,pcoordenadas pcoordenadas)** pergunta ao utilizador que tipo de jogada quer e agir corretamente depois da decisão. Caso seja um jogador e queira fazer uma jogada é chamada a função

**pjogadas pedeJogada(char\*\*tabuleiro,pdados,lista,robo,pcoordenadas)** que está encarregue das jogadas do jogador e que retorna a lista ligada do jogo com a jogada válida.

Caso queira consultar jogadas anteriores é chamada a função **void mostra\_info(pjogadas p,pjogadas pdados,int postjogadas,int robo)** que mostra a jogadas anteriores conforme o input do jogador.

A última opção para os jogadores é a interrupção do jogo que chama a função **void pause(pjogadas lista,pjogadas pdados)** que cria o ficheiro binário de acordo com a lista

ligada do jogo e de seguida é feita a libertação de memória ocupado pelo tabuleiro de jogo através do **void libertaMat(char\*\* p, int nLin)** e libertação da memória ocupada pela lista ligada **void elimina\_lista(pjogadas lista)**.

No caso de turno do computador a única opção que tem é ir para a função **pjogadas pedeJogada(char\*\*tabuleiro,pdados,lista,robo,pcoordenadas)**.

A função **pjogadas pedeJogada(char\*\*tabuleiro,pdados,lista,robo,pcoordenadas)** se for jogador pede a jogada e entra no ciclo “**while**” que usa a função **bool possiblePlay(char \*\*tabuleiro,pjogadas pdados,int bytes\_size,pcoordenadas pcoordenadas)** que enquanto for falso volta a pedir novamente a jogada. Para o computador entra no ciclo “**while**” semelhante que ao gerar uma random jogada, para validar a jogada é usa-se **bool converter\_coordenadas(char\*\* tabuleiro,pjogadas pdados, pcoordenadas pcoordenadas)** que converte as coordenadas locais para coordenadas globais do array bidimensional. Caso as jogadas sejam possíveis mostra como ficou o tabuleiro com print efetuado pela função **void mostraMat(char\*\* p, int nLin, int nCol)**, insere ordenadamente na lista a jogada através da função **pjogadas insere\_ord(pjogadas p,pjogadas pdados)**, a função **void nextquadro(pjogadas pdados,pcoordenadas pcoordenadas)** indica que mini-tabuleiro que se será utilizado na próxima jogada e no final retorna a lista com a jogada válida do programa.

### 2.3.5 tratar\_input.c

Em geral este ficheiro trata de converter as coordenadas locais atribuídas pelo jogador, e convertê-las para serem utilizadas no array bidimensional de jogo, decidir o próximo quadro a ser jogado, e validar o número de jogadas anteriores.

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

O quadro apresentado foi a maneira pensada para efetuar os cálculos de conversão, primeiro através do mini-tabuleiro em formato de 0 a 8 são geradas a posições em cima, uma para X e uma para Y, ao multiplicar essas por 3 e somar a jogada efetuada em formato 0 a 2 obtemos a coordenada para meter no array bidimensional do jogo. Caso seja uma jogada inválida retorna false para voltar a introduzir uma jogada, se for válida usa a função **bool jogador = checkTurnos(pdados)** para saber que símbolo meter no tabuleiro com a função **void setPos(char \*\*p, int x, int y, char c)** e retorna “true”.

Depois de uma jogada válida temos que calcular o mini-tabuleiro a jogar de seguida, aí entra a função **void nextquadro(pjogadas pdados,pcoordenadas pcoordenadas)** que usa a mesma filosofia da conversão de coordenadas só que ao contrário, que indica que quadro corresponde a jogada e atribui para a seguinte jogada. Se o mini-tabuleiro estiver completo, entra no ciclo “**while**” que enquanto não encontrar um mini-tabuleiro válido para jogar faz um random de todos os outros possíveis quadros.

**int jogadas\_anteriores(pjogadas pdados)** é usado para validar se a consulta de jogadas anteriores é possível. No primeiro turno simplesmente retorna 0 para a flag que não existe jogadas, caso existam jogadas com certas condições exigidas retorna o input feito pelo jogador.

### 2.3.5 winner.c

Ficheiro usado para detetar vitória no tabuleiro global e no tabuleiro local da seguinte forma.

**char check\_minitabuleiro(char\*\* tabuleiro,pjogadas pdados,pcoordenadas pcoordenadas)** faz a verificação se um tabuleiro local alcançou vitória ou empate, e retorna o símbolo associado a cada um.

**void arrayWinner(char\*\*tabuleiro,pjogadas pdados,pcoordenadas pcoordenadas)** chama a função anterior para apanhar o char retornado, e com isso preencher o **winnerArray** na posição correta com o char correspondente.

**char checkWinner(pcoordenadas pcoordenadas)** usa uma filosofia semelhante à da função **check\_minitabuleiro(char\*\* tabuleiro,pjogadas pdados,pcoordenadas pcoordenadas)** mas para este caso num array unidimensional **winnerArray** para verificação de vitória no tabuleiro global.

### 2.3.6 listas\_ligadas.c

Este ficheiro é como é ordenada a lista ligada, o seu preenchimento, e print total e parcial conforme a consulta pedida pelo jogador.

**pjogadas insere\_ord(pjogadas p,pjogadas pdados)** são criados duas estrutura de formato “**pjogadas**”, o “**novo**” e “**aux**” que vai ser necessário para inserir a lista na posição correta. A função **void preenche\_lista(pjogadas p,pjogadas pdados)** preenche a lista com dados necessários para o jogo e com struct “**prox**” a NULL.

Caso esteja vazia a lista insere na cabeça, caso contrário, guarda a lista num aux, avança até o final da lista, e insere essa lista por último, no final retorna a lista nova.

**void mostra\_info\_ex(pjogadas p)** imprime a lista ligada por completo.

**void mostra\_info(pjogadas p,pjogadas pdados,int postjogadas,int robo)** imprime a lista consoante a consulta de jogadas anteriores que pediu.

### 2.3.7 ficheiros.c

**bool ficheiro\_texto(pjogadas lista)** faz a exportação no final de um jogo ganho ou empatado para um ficheiro de texto com o nome desejado pelo o utilizador.

### 2.3.8 pausar.c

**void pause(pjogadas lista,pjogadas pdados)** que caso o jogador queira interromper o jogo a função guarda os dados adequadamente na lista. Se a lista estiver vazia, preencha com uma flag a 0 para ser o primeiro dado a ler. Caso contrário, a flag estará a 1 mais o valor dos turnos totais completados, e depois todos os dados necessários para recuperar o jogo.

### 2.3.8 recuperar\_jogo.c

Contém a verificação da existência de um binário, a resposta se o jogador quer recuperar o jogo, e por final a recuperação do jogo caso seja essa a decisão.

**bool verificarbin()** retorna “**false**” caso não exista binário e “**true**” se existir.

**bool querContinuar()** retorna “**false**” caso não recuperar o binário e “**true**” se o quiser recuperar.

**pjogadas recuperarJogo(pjogadas lista,pjogadas pdados,char**

**\*\*tabuleiro,pcoordenadas pcoordenadas)** abre o ficheiro binário, caso o dado a ser lido primeiro estiver com valor 0 avisa que não tem dados a recuperar e faz o jogo normal. Se estiver a 1, apanha o valor dos turnos que ocorreram, e lê o ficheiro binário no ciclo “**while**” até o valor dos **maxturnos** recebido chegar a 0. Dentro do “**while**” faz um fread para cada um dos dados guardados no ficheiro, volta a converter as coordenadas de cada jogada, chama as funções que fazem as verificações de vencedor nos tabuleiros local para preenchimento do **arrayWinner**, chama a função **pjogadas insere\_ord(pjogadas p,pjogadas pdados)** para fazer a lista novamente.

# 3. Manual de utilização

## 3.1 Menu

O jogador tem duas opções no menu principal (figura 3).

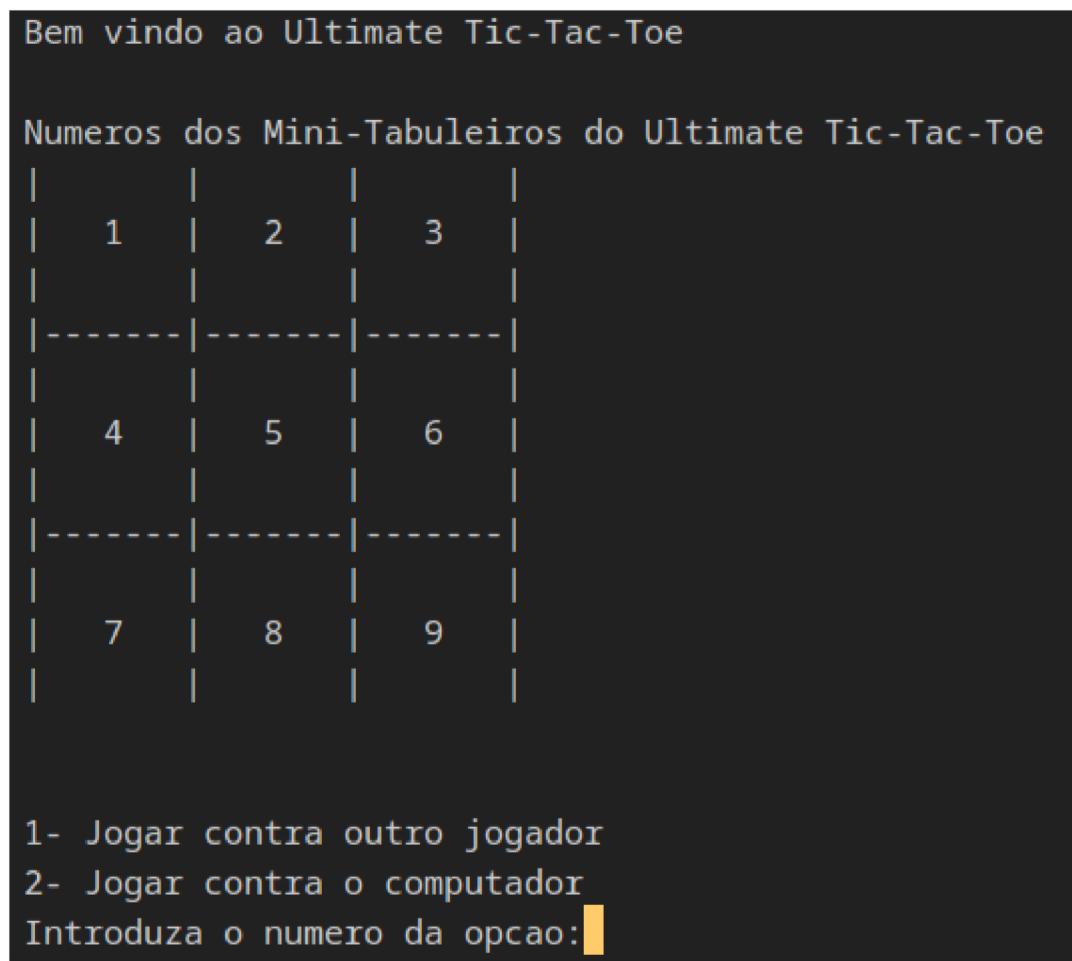


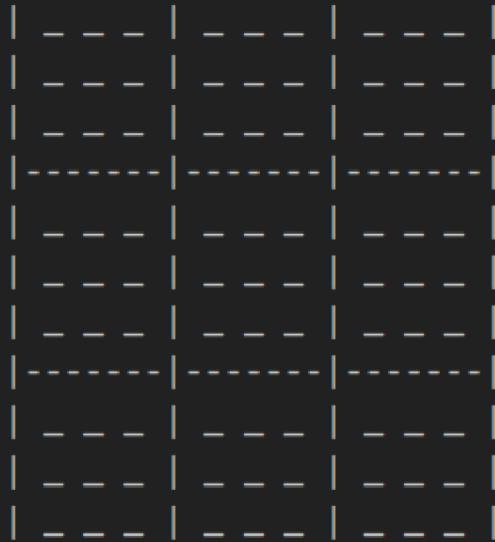
Figura 3

Ao escolher a primeira opção é necessário dois jogadores, na segunda opção é necessário só um.

## 3.2 Jogo

Após escolher uma das opções é criado um tabuleiro para os jogadores com estas possíveis jogadas (figura 4), este exemplo não contém o ficheiro binário.

```
Tabuleiro Inicial  
O jogo comeca no Mini-Tabuleiro 8
```



```
Nao existe jogo.bin
```

```
Jogador 1 a jogar || Simbolo X || Turno:1
```

```
Mini_tabuleiro 8
```

```
1- Fazer Jogada
```

```
2- Rever K jogadas anteriores(K < 10)
```

```
3- Sair do Jogo
```

```
Introduza o numero da opcao:
```

Figura 4

### 3.2.1 Fazer uma Jogada

Coloca uma peça na posição desejada, o número do mini-tabuleiro corresponde aos números associados logo no início do jogo (figura 3), o jogar desta forma pode pensar em cada tabuleiro localmente em que as jogadas podem ir de 1 a 3.

O formato de introdução é "Introduza o numero da opcao:X Y" o X é a linha, e Y a coluna (figura 5).

```
Introduza linha e coluna (ex:1 1,2 3):1 1
```

Figura 5

### 3.2.2 Rever uma Jogada ou mais Jogadas

O jogador que quiser rever jogadas seleciona a opção 2 da figura 4, que seguida pede ao utilizador quantas jogadas quer rever(figura 6).

```
Introduza k jogadas quer consultar:
```

Figura 6

Caso não existam jogadas anteriores, por exemplo no primeiro turno, são obrigados a fazer uma jogada.

Se houver jogadas, imprime as jogadas ordenadamente da última à mais recente(figura 7).

```
Jogador 2 a jogar || Simbolo 0 || Turno:8
Mini_tabuleiro 8
1- Fazer Jogada
2- Rever K jogadas anteriores(K < 10)
3- Sair do Jogo
Introduza o numero da opcao:2
Introduza k jogadas quer consultar:7

0 jogador 1 efetuou a jogada(x e y): 1 1 no mini-tabuleiro: 8 || turno 1
0 jogador 2 efetuou a jogada(x e y): 1 1 no mini-tabuleiro: 1 || turno 2
0 jogador 1 efetuou a jogada(x e y): 2 3 no mini-tabuleiro: 1 || turno 3
0 jogador 2 efetuou a jogada(x e y): 3 2 no mini-tabuleiro: 6 || turno 4
0 jogador 1 efetuou a jogada(x e y): 3 3 no mini-tabuleiro: 8 || turno 5
0 jogador 2 efetuou a jogada(x e y): 2 2 no mini-tabuleiro: 9 || turno 6
0 jogador 1 efetuou a jogada(x e y): 3 2 no mini-tabuleiro: 5 || turno 7
```

Figura 7

De seguida são obrigados a fazer uma jogada após a consulta.

### 3.2.3 Pausar o jogo

O jogador que quiser rever jogadas seleciona a opção 3 da figura 4, que avisa o utilizador se tem dados ou não, e cria um ficheiro binário adequadamente.(figura 7).

```
Escrever estado do jogo para jogo.bin... [paulitchos@theoverheatingmachine Trabalho_P]$
```

Figura 8

### 3.2.4 Vencedor

Caso haja vencedor ou empate pede ao utilizador para guardar o ficheiro em tipo “.txt” com nome desejado(figura 9) que inclui todas as jogadas feitas no jogo(figura 10).

```

Introduza linha e coluna (ex:1 1,2 3):1 3
| _ _ _ | _ _ _ | _ _ _ |
| 0 0 0 | 0 _ 0 | _ 0 _ |
| _ _ _ | _ _ _ | _ _ _ |
| -----|-----|-----|
| X X X | X _ _ | X X X |
| _ 0 _ | X _ _ | _ _ _ |
| _ _ _ | X _ _ | _ _ _ |
| -----|-----|-----|
| _ _ _ | _ _ _ | _ _ _ |
| 0 _ _ | _ _ _ | _ _ _ |
| _ _ _ | _ _ _ | _ _ _ |

O Jogador 1 ganhou Quadro 4
Jogador 1 ganhou o jogo || Simbolo X
Introduza o seu nome:SmoothJonnas

```

Figura 9

```

SmoothJonnas
|
0 jogador 1 efetuou a jogada(x e y): 1 1 no mini-tabuleiro: 6 || turno 1
0 jogador 2 efetuou a jogada(x e y): 2 3 no mini-tabuleiro: 1 || turno 2
0 jogador 1 efetuou a jogada(x e y): 1 2 no mini-tabuleiro: 6 || turno 3
0 jogador 2 efetuou a jogada(x e y): 2 3 no mini-tabuleiro: 2 || turno 4
0 jogador 1 efetuou a jogada(x e y): 1 3 no mini-tabuleiro: 6 || turno 5
0 jogador 2 efetuou a jogada(x e y): 2 2 no mini-tabuleiro: 3 || turno 6
0 jogador 1 efetuou a jogada(x e y): 1 1 no mini-tabuleiro: 5 || turno 7
0 jogador 2 efetuou a jogada(x e y): 2 2 no mini-tabuleiro: 1 || turno 8
0 jogador 1 efetuou a jogada(x e y): 2 1 no mini-tabuleiro: 5 || turno 9
0 jogador 2 efetuou a jogada(x e y): 2 2 no mini-tabuleiro: 4 || turno 10
0 jogador 1 efetuou a jogada(x e y): 3 1 no mini-tabuleiro: 5 || turno 11
0 jogador 2 efetuou a jogada(x e y): 2 1 no mini-tabuleiro: 7 || turno 12
0 jogador 1 efetuou a jogada(x e y): 1 1 no mini-tabuleiro: 4 || turno 13
0 jogador 2 efetuou a jogada(x e y): 2 1 no mini-tabuleiro: 1 || turno 14
0 jogador 1 efetuou a jogada(x e y): 1 2 no mini-tabuleiro: 4 || turno 15
0 jogador 2 efetuou a jogada(x e y): 2 1 no mini-tabuleiro: 2 || turno 16
0 jogador 1 efetuou a jogada(x e y): 1 3 no mini-tabuleiro: 4 || turno 17

```

Figura 10

### 3.3 Recuperação do Jogo Anterior

Caso ele detecte um jogo.bin, isto é, o jogo anteriormente interrompido, pergunta ao utilizador se o deseja recuperá-lo (figura 11).

```
Tabuleiro Inicial
O jogo começa no Mini-Tabuleiro 1

| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| -----|-----|-----|
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| -----|-----|-----|
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
Quer continuar o jogo anterior(Y | N)Introduza o opção:
```

Figura 11

Caso o utilizador diga que não, o jogo novo decorre normalmente.

Se o utilizador disser que sim é feita a recuperação do jogo anterior(figura 12).

```
Quer continuar o jogo anterior(Y | N):Y

Tabuleiro de Jogo Recuperado
| 0 _ _ | _ _ - | _ _ - |
| _ _ X | _ _ - | _ _ - |
| _ _ - | _ _ - | _ _ - |
| -----| -----| -----|
| _ _ - | _ _ - | _ _ - |
| _ _ - | _ _ - | _ _ - |
| _ _ - | _ X _ | _ 0 _ |
| -----| -----| -----|
| _ _ - | X 0 _ | _ _ - |
| _ _ - | _ _ - | _ 0 _ |
| _ _ - | _ _ X | _ _ - |

Jogador 1 a jogar || Simbolo X || Turno:9
```

Figura 12

Caso ele diz que sim mas o bin vazio, avisa o utilizador e efetua um jogo novo(figura 13)

```
Quer continuar o jogo anterior(Y | N):Y

Nao tem dados para recuperar, recomendar jogo
```

Figura 13