Programação
Relatório - Jogo do Semáforo

Docente: Francisco Pereira
Trabalho realizado por: Nelson Simão nº2020132648- LEI

Indíce

1 Principais estruturas de dados

No programa existem 5 estruturas de dados principais: struct plays, struct player, struct coordinates, struct list_head e struct list_node.

1.1 struct plays

```
struct plays{ //cada membro representa uma jogada que um jogador pode realizar
  bool green;
  bool yellow;
  bool red;
  bool rock;
  int lc;
  bool k_interrupt;
};
```

Esta estrutura serve para indicar que jogadas um jogador pode realizar ao longo do jogo.

- **bool green** - toma o valor 1 se o jogador puder colocar uma peça verde no tabuleiro, 0 se nao o puder fazer
- **bool yellow** - toma o valor 1 se o jogador puder trocar uma peça verde por uma amarela no tabuleiro, 0 se nao o puder fazer
- **bool red** - toma o valor 1 se o jogador puder trocar uma peça verde por uma amarela no tabuleiro, 0 se nao o puder fazer
- **bool rock** - toma o valor 1 se o jogador puder colocar uma pedra no tabuleiro, 0 se nao o puder fazer, um jogador só pode realizar esta jogada uma vez por jogo
- **int lc** - numero de colunas ou linhas que o jogador pode inserir no tabuleiro, um jogador só pode realizar esta jogada 2 vezes por jogo
- **bool k_interrupt** - toma o valor 1 se o jogador puder ver as k jogadas anteriores ou se puder interromper o jogo, ambas jogadas só podem ser feitas após o 1º turno

1.2 struct player

```
struct player{ //estrutura jogador
  char name;
  struct plays ability;
};
```

Estrutura que representa um jogador. Contem outra estrutura(**struct plays**) que representa as jogadas que o jogador pode realizar, como já foi mencionado anteriormente. **char name** representa o nome do jogador, 'A' ou 'B'.

1.3 struct coordinates

```
struct coordinates{ //coordenadas de uma localizacao do tabuleiro
  int x,y;
};
```

Esta estrutura serve para guardar as coordenadas de uma celula do tabuleiro, (x,y).

1.4 struct list_node

```
struct list_node{ //nos da lista, onde informacoes sobre as jogadas realizadas estarao
  int turn;
  int lin,col;
  char piece;
  struct coordinates place;
  char player_name;
  struct list_node *next;
};
```

Para implementar a funcionalidade de "visualizar o estado do tabuleiro nas K jogadas anteirores" e ainda a "exportação para um ficheiro de texto", era obrigatório a implementação de uma lista ligada. A estrutura **struct list_node** representa um nó desta lista ligada. O objetivo desta lista ligada é armazenar informação sobre as jogadas realizadas ao longo do jogo.

- **int turn** - Turno atual, representa tambem o indice do nó na lista
- **int lin** - Número de linhas do tabuleiro no turno atual
- **int col** - Número de linhas do tabuleiro no turno atual
- **char piece** - Carater que representa a peça/jogada realizada nesse turno(se for esse o caso, ex: se **piece**=='C' foi adicionada uma coluna e nenhum carater foi colocado no tabuleiro)
- **struct coordinates place** - Coordenadas do tabuleiro em que a peça foi colocada
- **struct list_node *next** - Ponteiro para o próximo nó

1.5 struct list_head

```c
struct list_head{ //cabeca da lista, onde o tabuleiro inicial estara
  char **tab;
  int lin,col;
  struct list_node *next;
};
```

A estrutura **struct list_head** representa a cabeça da lista ligada. A lista ligada implementada neste programa é uma "lista ligada mista", ou seja, a cabeça da lista é diferente dos seus nós. A lista ligada será explicada na secção 2.

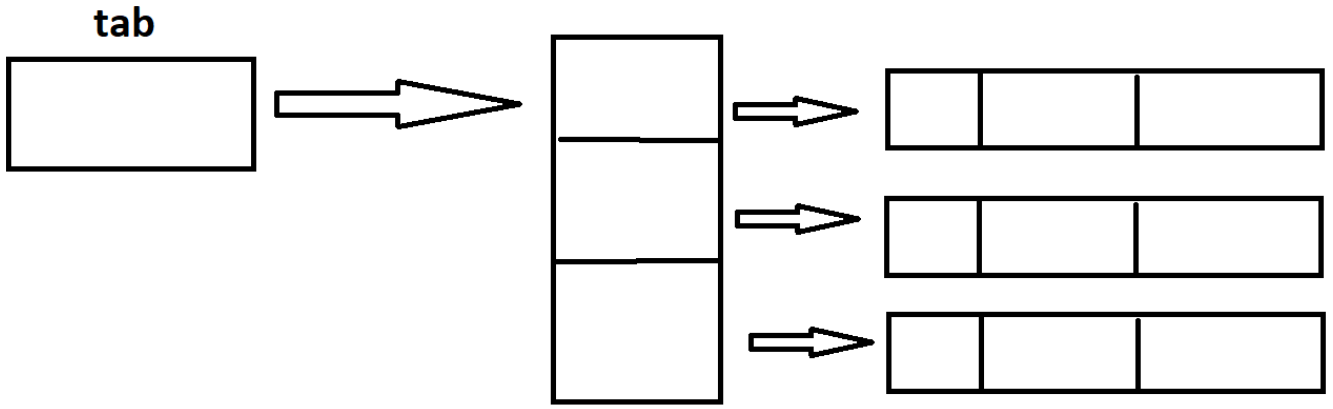
- **char **tab** - Este ponteiro para ponteiro aponta para um array bidimensional alocado dinamicamente, e representa o tabuleiro do jogo no inicio do jogo.
- **int lin** - Número de linhas de **tab**
- **int col** - Número de colunas de **tab**
- **struct list_node *next** - ponteiro para um nó da lista

2 Estruturas dinâmicas implementadas

Este programa contém duas estruturas dinâmicas: o array bidimensional que representa o tabuleiro de jogo e a lista ligada que armazena informação sobre as jogadas realizadas ao longo do jogo.

2.1 char **tab

Esta estrutura dinâmica, um vetor bidimensional alocado dinamicamente, representa o tabuleiro do jogo e, sendo que é alocado dinamicamente, é possivel a adição de linhas e colunas. Segue-se um esquema do tabuleiro com 3 linhas e 3 colunas:



É possível observar então que **tab** aponta para um array alocado dinamicamente de **char *** e cada um dos elementos deste array aponta para outro array alocado dinamicamente de **char**. Através de simples aritmética de ponteiros podemos perceber que podemos aceder ao elemento da linha **i**, coluna **j**, com **tab[i][j]**. Escusado dizer que um acesso tao simples a um elemento permite uma grande simplificacão das operações no tabuleiro. Este fator, aliado à simplicidade da estrutura de dados(que no fundo nao difere muito de um array bidimensional alocado na stack) foi o motivo pelo qual optei por esta estrutura de dados para representar o tabuleiro de jogo.

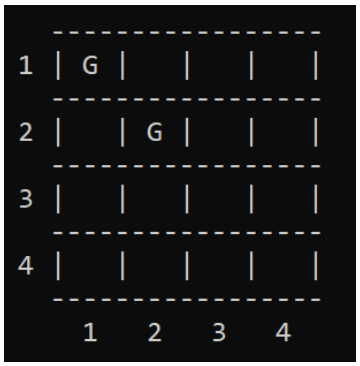
2.2 lista ligada

A lista ligada serve para um jogador, puder visualizar o estado do tabuleiro nas **K** jogadas anteriores e ainda para poder exportar a sucessão de estados do tabuleiro para um ficheiro .txt.

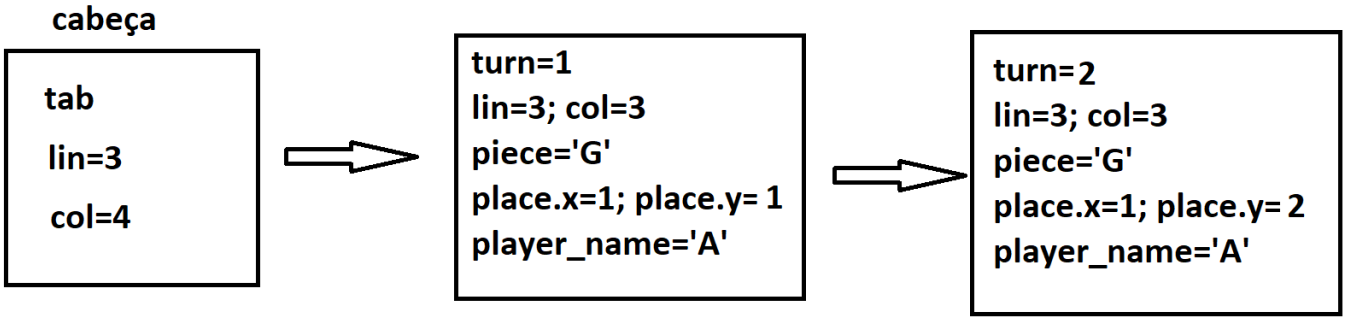
A lista ligada por mim escolhida é uma lista ligada "mista", ou seja, a cabeça da lista é diferente dos nós que a compõem. A cabeça da lista irá armazenar o tabuleiro vazio, tal e qual como ele estava no principio do jogo(quadrado e com um numero aleatório de linhas) e irá armazenar também as suas dimensões (linhas e colunas). Cada nó da lista, que é de um tipo diferente da cabeça, corresponde a um turno, e cada nó, armazena informação sobre a jogada realizada nesse turno(ver definição de um nó na secção anterior). O objetivo desta separação é muito simples.

Segue-se agora um exemplo que mostra a utilidade da lista, não pretendo explicar todas as funções subjacentes às operações que irei mostrar, pretendo é exemplificar um exemplo de aplicação da lista ligada.

Suponhamos então que estamos no turno 3, vez do jogador A, e o tabuleiro encontra-se no seguinte estado:



A lista encontra-se neste estado:



Na cabeça **tab** está a apontar para um tabuleiro alocado dinamicamente de dimensao 4x4.

Quando o jogador A seleciona a opção de ver as 2 jogadas anteriores, é chamada uma função que conforme a informação em cada nó, vai alterando o tabuleiro que está na cabeça da lista até chegar ao nó com **turn**=2, e enquanto atualiza o tabuleiro que esta na cabeça da lista, vai escrevendo esse tabuleiro e informação sobre a jogada realizada nesse turno na consola. Para este exemplo obtemos então:

```
As 2 jogadas anteriores foram:

Turno 1
O jogador A colocou a peca G na posicao 1 1
    -----------------
  1 | G |   |   |   |
    -----------------
  2 |   |   |   |   |
    -----------------
  3 |   |   |   |   |
    -----------------
  4 |   |   |   |   |
    -----------------
      1   2   3   4

Turno 2
O jogador B colocou a peca G na posicao 2 2
    -----------------
  1 | G |   |   |   |
    -----------------
  2 |   | G |   |   |
    -----------------
  3 |   |   |   |   |
    -----------------
  4 |   |   |   |   |
    -----------------
      1   2   3   4
```

No final de toda esta operação, o tabuleiro presente na cabeça volta ao estado em que estava no inicio do jogo. Com uma lista assim fui capaz de mostrar informação completa e detalhada sobre a sucessão de estados do tabuleiro na consola, e também, no ficheiro de texto(o procedimento é semelhante mas mais linear). O facto de poder escrever informação detalhada sobre as jogadas realizadas, foi o meu motivo de escolha desta lista ligada.

3 Funcionamento do Programa

No inicio do programa, a função **main** apresenta um menu ao utilizador que lhe permite escolher se, quer jogar contra outro jogador, se quer jogar contra o computador, se quer ver as regras do jogo ou se quer sair do programa. Nesta secção irei-me focar nas opções de jogo já que as duas outras são muito simples. A função **main** apenas tem o propósito de receber input do utilizador, validá-lo, e conforme a opção escolhida, supondo que o utilizador quer jogar, chamar a função **game(bool game_mode, bool resume**.

O jogo decorre na função **game** onde:

- **bool game_mode** - 0 indica que o jogo deverá se realizar a dois jogadores humanos, 1 indica que o jogo será contra o computador
- **bool resume** - 0 indica que é para começar um jogo novo, 1 indica que é para continuar um jogo anterior

Se o utilizador não quiser continuar o jogo anteriror e escolher a opção de jogar contra o computador, a função **main** chamará a função **game** com **game_mode=1, resume=0**.

Se o ficheiro **jogo.bin** existir, quer dizer que há um jogo por retomar, e nesse caso a função main apenas pergunta ao utilizador ser quer continuar o jogo. Se o utilizador nao quiser então o menu normal ser-lhe-à apresentado, caso contrário **main** acede a **jogo.bin**, determina o modo de jogo e invoca a função **game** com os argumentos apropriados.

Quando o controlo passa para **game**, esta função dependendo dos parametros com que foi chamada inicia ou retoma um jogo. Por exemplo: se a função game foi chamada com **game_mode=0, resume=1**, então a função chama outras funções auxiliares que inicializão as variaveis necessárias ao funcionamento do jogo com os valores que elas tinham quando o jogo foi interrompido e, se essa operação for um sucesso, o jogo continua normalmente. Se o jogo for contra o computador, por exemplo, em vez de **game** pedir uma jogada ao jogador 'B', pede uma jogada ao "jogador autmático".

O jogo então sucede normalmente, a pedir jogadas a ambos os jogadores alternadamente, e o tabuleiro de jogo vai sendo alterado. No final de um turno é adicionado um nó à lista ligada, com informação sobre as jogadas realizadas nesse turno. O jogo acaba quando um dos jogadores vencer, vence o jogador que conseguir formar uma linha, coluna ou diagonal com peças da mesma cor. Quando isso acontece é pedido ao utilizador um nome para um ficheiro .txt onde informação sobre o jogo ficará armazenada. Após isto o programa termina.

4 Pequeno Manual de Utilização

Assim que o programa começa é apresentado o seguinte menu ao utilizador:

```
========== Jogo do semaforo ==========

Bem vindo ao jogo do semaforo!

Escolha uma opcao:

Opcao 1: Jogar contra outro jogador(localmente)
Opcao 2: Jogar contra o computador
Opcao 3: Regras
Opcao 4: Sair

Opcao:
```

Se o utilizador escolher a opção 3, as regras do jogo são apresentadas na consola e ainda é dada a possibilidade de as regras serem guardadas num ficheiro .txt para poderem ser consultadas a meio do jogo. No final de as regras estarem mostradas o menu volta a aparecer.

Independentemente da opção de jogo escolhida pelo utilizador, opção 1 ou opção 2, o resultado será sempre o mesmo:

```
Turno 1, vez do jogador A

    --------------------
  1 |   |   |   |   |   |
    --------------------
  2 |   |   |   |   |   |
    --------------------
  3 |   |   |   |   |   |
    --------------------
  4 |   |   |   |   |   |
    --------------------
  5 |   |   |   |   |   |
    --------------------
      1   2   3   4   5

As jogadas que pode fazer sao:
Adicionar uma peca verde numa celula vazia (G)
Colocar uma pedra numa celula vazia (S) (1 restantes)
Adicionar uma linha ou coluna (L ou C) (2 restantes)


Digite uma jogada das listadas acima:
```

O jogador A é sempre um jogador humano e é o primeiro a começar. É apresentado um desenho do tabuleiro, incialmente vazio, e as jogadas que o jogador atual pode fazer são apresentadas no ecrã, com uma descrição sobre a jogada e o carater entre "()" associado a essa jogada.

Para jogar basta então pressionar um carater válido:

```
Digite uma jogada das listadas acima: G
```

Se for o caso, como neste exemplo, inserir as coordenadas em que a peça deve ser colocada no tabuleiro:

```
Digite as coordenadas do tabuleiro em que quer inserir a peca: 1 2
```

E após isso o tabuleiro será atualizado, o turno incrementado, e passamos para a vez do jogador B.

```
Turno 2, vez do jogador B

    -----------------
  1 |   | G |   |   |
    -----------------
  2 |   |   |   |   |
    -----------------
  3 |   |   |   |   |
    -----------------
  4 |   |   |   |   |
    -----------------
      1   2   3   4

As jogadas que pode fazer sao:
Adicionar uma peca verde numa celula vazia (G)
Trocar uma peca verde por uma amarela (Y)
Colocar uma pedra numa celula vazia (S) (1 restantes)
Adicionar uma linha ou coluna (L ou C) (2 restantes)
Ver as k jogadas anteriores(K)
Interromper o jogo para ser ou nao retomado posteriormente(I)


Digite uma jogada das listadas acima:
```

Podemos ver que o jogador B já pode realizar jogadas que o jogador A nao pôde realizar. Isto deve-se a uma peça verde ter sido inserida no tabuleiro o que implica que o jogador B poderá então trocar essa peça por uma amarela. Também podemos verifcar que o jogador B já pode interromper o jogo e ver as K jogadas anteriores, isto deve-se ao facto de estarmos no turno 2. Não faria muito sentido interromper o jogo no primeiro turno nem era possivel visualizar qualquer jogada anterior no primeiro turno.

Qualquer tentativa de inserir um carater invalido, um carater que representa uma jogada que o jogador nao pode realizar no momento ou umas coordenadas inválidas, serão bloqueadas e uma mensagem de erro será exibida a pedir ao utilizador que insira esses dados corretamente.

```
Digite uma jogada das listadas acima: R
Atualmente nao pode fazer essa jogada
Por favor digite uma jogada das listadas acima: G
Digite as coordenadas do tabuleiro em que quer inserir a peca: -3 6
Digite coordenadas validas: ola
Digite as coordenadas no formato certo x y:
```

Se escolher o carater 'I', será exibida uma mensagem a indicar se foi possível guardar informação para retomar o jogo num ficheiro binário, ou uma mensagem de erro se tal operação nao foi possível, em ambos os casos o programa terminará.

No final do jogo, vitória de um jogador, é apresentado o tabuleiro atualizado e uma mensagem a indicar que jogador venceu:

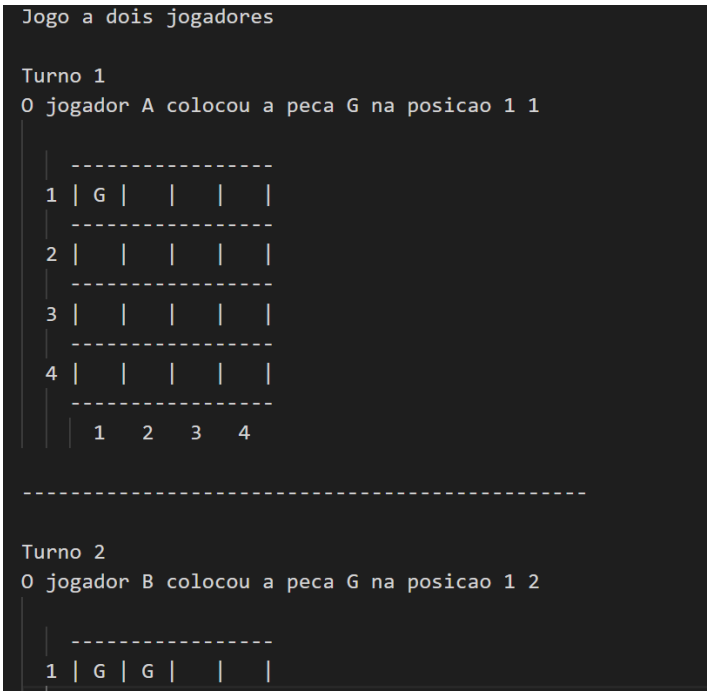
```
O jogador A venceu o jogo!!!

    -----------------
  1 | G | G | G | G |
    -----------------
  2 |   |   |   |   |
    -----------------
  3 |   |   |   |   |
    -----------------
  4 |   |   |   | S |
    -----------------
      1   2   3   4

Digite o nome do ficheiro em que quer que o seu jogo seja guardado, insira um nome com 10 carateres no maximo:
```

E é pedido ao utilizador o nome para um ficheiro que irá guardar a sucessão de estados do tabuleiro. São lidos no máximo 10 carateres e a extensão .txt é adicionada pelo programa.

Segue-se uma imagem do ficheiro .txt:



O jogo contra o computador não será ilustrado nesta secção. A única diferença deste modo de jogo é que o jogador A é o utilizador, e este joga contra o computador. Vale a pena salientar que o jogador automático realiza uma jogada válida aleatoriamente e que nunca irá interromper o jogo ou visualizar as K jogadas anteriores. De resto comporta-se como um jogador humano normal.