_____ / _____ / _____

Gerador de movimentos de xadrez

O xadrez tem um papel fundamental na Inteligência Artificial, e por conseguinte na ciência da computação. Ele é suficientemente compacto para poder ser manuseado dentro de um computador, mesmo um computador da década de 50, com poucas centenas de bytes de memória. Mas, é abrangente e complexo para representar o grau máximo de abstração e simbolismo de que o ser humano é capaz. Turing, o genial criador desta ciência, sugeriu em 1951 que *"em 50 anos, um computador será capaz de vencer o campeão mundial de xadrez"*. Ele errou por pouco: 1951+50=2001. Em 1997, Deep Blue, da IBM, venceu Gary Kasparov que era o campeão mundial. Que outra previsão – no mundo da computação – tem, teve ou terá tal grau de acerto ? Já foi chamado de *drosóphila melanogaster* da IA, similar à importância da mosca doméstica na genética.

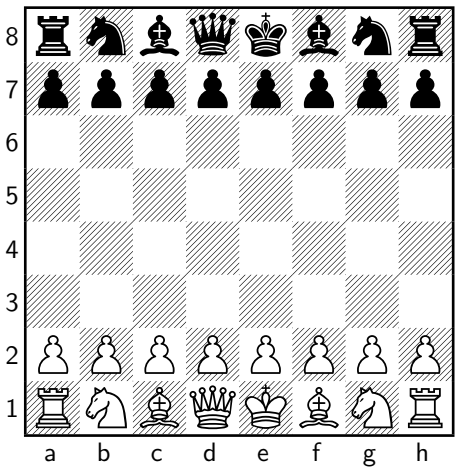
Abstraindo a computação, ainda assim o xadrez é um mundo maravilhoso. Aqui no Brasil, não temos grande tradição nele, mas em uma boa parte do mundo, as crianças aprendem a jogar e jogam durante toda sua vida escolar e por conseguinte muitos adultos seguem jogando vida afora. Isto, desde o século XII, que registra as primeiras manifestações do jogo, que por sinal é muito mais antigo do que isso.

Ele representa 2 exércitos que se opõe em um tabuleiro limitado (e pequeno) de 64 casas, e o único objetivo do jogo é vencer o exército inimigo. Esse fato é representado pela captura do rei oposto. Compõe o exército, as forças usuais: a infantaria, representada pelos peões em número de 8. As máquinas de guerra, as torres em número de 2. A igreja, personagem sempre influente, pelos seus 2 bispos. A cavalaria com 2 cavalos. Finalmente o casal real: a raínha ou dama, a peça mais poderosa no tabuleiro e o rei, limitado em movimentos, mas a peça em volta da qual todos operam.

Jogar xadrez é uma viagem intelectual indescritível, puro prazer. É superado pela construção de um programa de computador que jogue xadrez. Não é tarefa simples, mas deveria ser um investimento intelectual de todos quantos pretendam ser programadores plenos de computadores.

Como sempre, um programa grande e complexo precisa ser dividido em funções menores em tamanho e escopo. Uma parte importante em qualquer jogador é a análise e descoberta de todas as jogadas possíveis de uma determinada peça. Este exercício vai pedir que você programe uma pequena parte desta função.

Preliminares Antes de começar, é preciso se familiarizar com um tabuleiro armado.



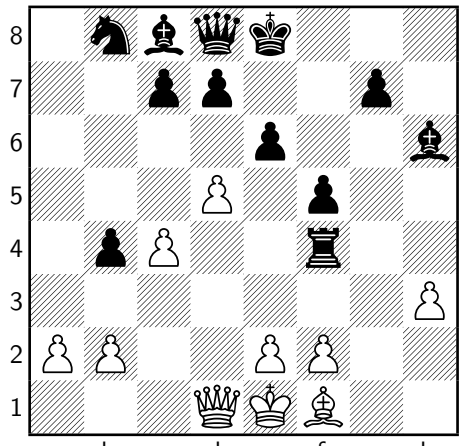
Tal disposição é boa para humanos, mas computadores precisam outra coisa: uma matriz numérica de 8 × 8, com casa brancas e negras alternadas sendo que que a casa embaixo, mais à direita, é branca. Por ser numérica, a matriz usa números para identificar as peças. Primeiro, a cor. Peças brancas são positivas e negras, negativas. Os valores: 1=peão, 2=torre, 3=cavalo, 4=bispo, 5=dama e 6=rei. Note que a célula linha 0, coluna 0, é a superior esquerda da matriz. Nesses termos, o tabuleiro acima corresponde a

	0	1	2	3	4	5	6	7
0|	-2	-3	-4	-5	-6	-4	-3	-2
1|	-1	-1	-1	-1	-1	-1	-1	-1
2|	0	0	0	0	0	0	0	0
3|	0	0	0	0	0	0	0	0
4|	0	0	0	0	0	0	0	0
5|	0	0	0	0	0	0	0	0
6|	1	1	1	1	1	1	1	1
7|	2	3	4	5	6	4	3	2

Você deve escrever um programa que vai receber um tabuleiro preenchido com uma situação de meio-jogo e também uma linha (0-7) e coluna (0-7). Você deve olhar o tabuleiro fornecido e ver qual é a peça que está lá. Identificada a peça e a sua cor, deve descobrir quantas casas essa peça pode ocupar num eventual próximo lance. Lembrando que as casas examinadas:

- se contiverem zero, podem ser ocupadas.
- se contiverem valor de sinal oposto ao da peça podem ser ocupadas, mas este valor implica parar na busca nessa direção (exceto o cavalo).
- se contiverem valor de mesmo sinal, não podem ser ocupadas e determinam final da busca.
- A casa não deve ser considerada se algum índice (linha ou coluna ou ambos) cair fora do tabuleiro.

Finalmente, não se assuste se aparecer um tabuleiro com alguma situação muito infrequente do ponto de vista enxadrístico: o gerador usa e abusa de números aleatórios. Vejamos um exemplo



Se o endereço a pesquisar fosse 4,5 a resposta deveria ser 7. Se fosse 1,2 a resposta deveria ser 2. Se fosse 0,1 a resposta deveria ser 2.

Movimento do peão Se na casa inicial, pode adiantar 1 ou 2 casas. Se em outras casas, só pode andar 1 posição. Pode tomar em diagonal, à direita ou à esquerda. Não se implementa aqui a tomada *en passant*, pois ela depende da jogada anterior, que neste caso é desconhecida.

```
int peao(int x[8][8], int lin, int col){
  int casa = 0;
  if (x[lin][col]==1) {
    if (lin > 0 && x[lin-1][col]==0) {
      casa++;      }
    if (lin==6 &&x[4][col]==0 && x[5][col]==0){
      casa++;     }
    if (lin>0 && col<7 &&x[lin-1][col+1]<0) {
      casa++;         }
    if (lin>0 && col>0 &&x[lin-1][col-1]<0) {
      casa++;          }       }
  else {
    if (lin <7 && x[lin+1][col]==0) {
      casa++;       }
    if (lin==1 &&x[2][col]==0 && x[3][col]==0){
      casa++;         }
    if (lin<7 && col<7 &&x[lin+1][col+1]<0) {
      casa++;         }
    if (lin<7 && col>0 &&x[lin+1][col-1]<0) {
      casa++;         }      }
  return  casa;     }
```

Torre

```
int torre(int x[8][8], int lin, int col){
  int casa = 0;
  int nl = lin-1;
  int nc =col;
  while (nl>=0){
    if (x[nl][nc]*x[lin][col]==0) {
      casa++;  }
    if (x[nl][nc]*x[lin][col]<0) {
      casa++; nl=0; }
    if (x[nl][nc]*x[lin][col]>0) {
      nl=0; }
      nl++; }
    nl=lin+1;
  // ... nl<= 7...
  // ... nc<= 7...
  // ... nc>= 7...
  return casa;}
```

Cavalo

```
int cavalo(int x[8][8], int lin, int col){
  int i,nl,nc,casa = 0;
  int dlin[8]={-2,-1,1,2,2,1,-1,-2};
  int dcol[8]={1,2,2,1,-1,-2,-2,-1};
  for (i=0;i<8;i++){
    nl=lin+dlin[i];
    nc=col+dcol[i];
    if (nl<=7 && nl>=0 && nc<=7 && nc>=0){
      if (x[nl][nc]==0||x[nl][nc]*x[lin][col]<0){
        casa++; }  }   }
  return casa;
```

Bispo

```
int bispo(int x[8][8], int lin, int col){
  int nl,nc,casa=0;
  nl=lin-1;
  nc=col-1;
  while (nl>=0 && nc>=0){
    if (x[nl][nc]==0) {
      casa++; }
    if (x[nl][nc]*x[lin][col]<0){
      casa++;   nl=0;}
    if (x[nl][nc]*x[lin][col]>0){
      nl=0; }
    nl--;
    nc--;  }
  nl=lin-1;
  nc=col+1;
  // while nl>=0 && nc<=7...
  // while nl<=7 &&nc >=0...
  // while nl<=7 && nc<=7...
  return casa;       }
```

Dama e Rei Fazendo os devidos ajustes aos controles de torre e bispo, é quase trivial. Ficam para você fazer, por falta de espaço.

☞ Para você fazer

A seguir um tabuleiro em situação de meio jogo, seguido do endereço de 3 peças. Escreva um programa capaz de descobrir quantas jogadas cada peça individualizada pode realizar na jogada a seguir.

	0	1	2	3	4	5	6	7
0|	0	0	0	0	-6	0	0	0
1|	0	0	0	0	-1	0	0	0
2|	0	4	0	-1	0	-1	-1	-4
3|	0	0	0	1	0	5	0	-1
4|	0	1	0	0	1	0	4	0
5|	0	0	0	0	0	0	2	-2
6|	1	0	0	0	0	1	-3	0
7|	0	3	0	0	6	-3	0	0

Descubra quantas jogadas podem ser feitas pelas peças localizadas em: 1) 2 1 , 2) 4 6 e 3) 2 3 .

Responda aqui:

1	2	3

Gerador de movimentos de xadrez

O xadrez tem um papel fundamental na Inteligência Artificial, e por conseguinte na ciência da computação. Ele é suficientemente compacto para poder ser manuseado dentro de um computador, mesmo um computador da década de 50, com poucas centenas de bytes de memória. Mas, é abrangente e complexo para representar o grau máximo de abstração e simbolismo de que o ser humano é capaz. Turing, o genial criador desta ciência, sugeriu em 1951 que *"em 50 anos, um computador será capaz de vencer o campeão mundial de xadrez"*. Ele errou por pouco: 1951+50=2001. Em 1997, Deep Blue, da IBM, venceu Gary Kasparov que era o campeão mundial. Que outra previsão – no mundo da computação – tem, teve ou terá tal grau de acerto ? Já foi chamado de *drosóphila melanogaster* da IA, similar à importância da mosca doméstica na genética.

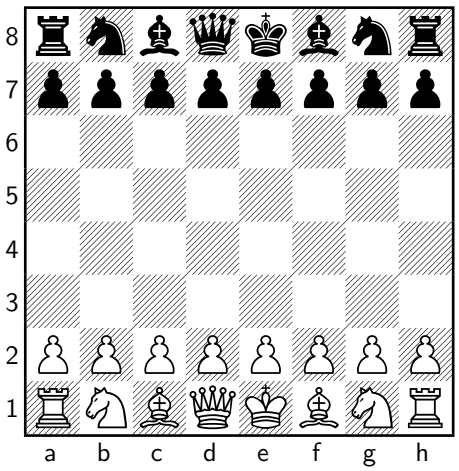
Abstraindo a computação, ainda assim o xadrez é um mundo maravilhoso. Aqui no Brasil, não temos grande tradição nele, mas em uma boa parte do mundo, as crianças aprendem a jogar e jogam durante toda sua vida escolar e por conseguinte muitos adultos seguem jogando vida afora. Isto, desde o século XII, que registra as primeiras manifestações do jogo, que por sinal é muito mais antigo do que isso.

Ele representa 2 exércitos que se opõe em um tabuleiro limitado (e pequeno) de 64 casas, e o único objetivo do jogo é vencer o exército inimigo. Esse fato é representado pela captura do rei oposto. Compõe o exército, as forças usuais: a infantaria, representada pelos peões em número de 8. As máquinas de guerra, as torres em número de 2. A igreja, personagem sempre influente, pelos seus 2 bispos. A cavalaria com 2 cavalos. Finalmente o casal real: a raínha ou dama, a peça mais poderosa no tabuleiro e o rei, limitado em movimentos, mas a peça em volta da qual todos operam.

Jogar xadrez é uma viagem intelectual indescritível, puro prazer. É superado pela construção de um programa de computador que jogue xadrez. Não é tarefa simples, mas deveria ser um investimento intelectual de todos quantos pretendam ser programadores plenos de computadores.

Como sempre, um programa grande e complexo precisa ser dividido em funções menores em tamanho e escopo. Uma parte importante em qualquer jogador é a análise e descoberta de todas as jogadas possíveis de uma determinada peça. Este exercício vai pedir que você programe uma pequena parte desta função.

Preliminares Antes de começar, é preciso se familiarizar com um tabuleiro armado.



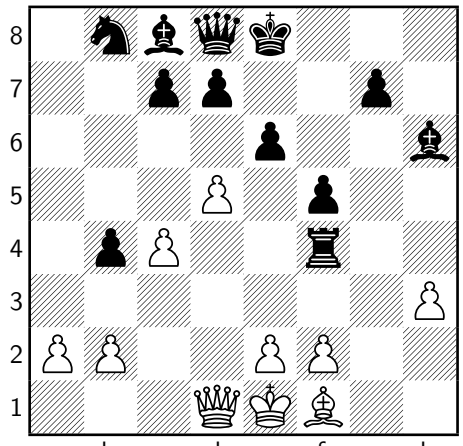
Tal disposição é boa para humanos, mas computadores precisam outra coisa: uma matriz numérica de 8 × 8, com casa brancas e negras alternadas sendo que que a casa embaixo, mais à direita, é branca. Por ser numérica, a matriz usa números para identificar as peças. Primeiro, a cor. Peças brancas são positivas e negras, negativas. Os valores: 1=peão, 2=torre, 3=cavalo, 4=bispo, 5=dama e 6=rei. Note que a célula linha 0, coluna 0, é a superior esquerda da matriz. Nesses termos, o tabuleiro acima corresponde a

```
     0   1   2   3   4   5   6   7
  +-------------------------------
0|  -2  -3  -4  -5  -6  -4  -3  -2
1|  -1  -1  -1  -1  -1  -1  -1  -1
2|   0   0   0   0   0   0   0   0
3|   0   0   0   0   0   0   0   0
4|   0   0   0   0   0   0   0   0
5|   0   0   0   0   0   0   0   0
6|   1   1   1   1   1   1   1   1
7|   2   3   4   5   6   4   3   2
```

Você deve escrever um programa que vai receber um tabuleiro preenchido com uma situação de meio-jogo e também uma linha (0-7) e coluna (0-7). Você deve olhar o tabuleiro fornecido e ver qual é a peça que está lá. Identificada a peça e a sua cor, deve descobrir quantas casas essa peça pode ocupar num eventual próximo lance. Lembrando que as casas examinadas:

- se contiverem zero, podem ser ocupadas.
- se contiverem valor de sinal oposto ao da peça podem ser ocupadas, mas este valor implica parar na busca nessa direção (exceto o cavalo).
- se contiverem valor de mesmo sinal, não podem ser ocupadas e determinam final da busca.
- A casa não deve ser considerada se algum índice (linha ou coluna ou ambos) cair fora do tabuleiro.

Finalmente, não se assuste se aparecer um tabuleiro com alguma situação muito infrequente do ponto de vista enxadrístico: o gerador usa e abusa de números aleatórios. Vejamos um exemplo



Se o endereço a pesquisar fosse 4,5 a resposta deveria ser 7. Se fosse 1,2 a resposta deveria ser 2. Se fosse 0,1 a resposta deveria ser 2.

Movimento do peão Se na casa inicial, pode adiantar 1 ou 2 casas. Se em outras casas, só pode andar 1 posição. Pode tomar em diagonal, à direita ou à esquerda. Não se implementa aqui a tomada *en passant*, pois ela depende da jogada anterior, que neste caso é desconhecida.

```
int peao(int x[8][8], int lin, int col){
  int casa = 0;
  if (x[lin][col]==1) {
    if (lin > 0 && x[lin-1][col]==0) {
      casa++;      }
    if (lin==6 &&x[4][col]==0 && x[5][col]==0){
      casa++;     }
    if (lin>0 && col<7 &&x[lin-1][col+1]<0) {
      casa++;         }
    if (lin>0 && col>0 &&x[lin-1][col-1]<0) {
      casa++;          }       }
  else {
    if (lin <7 && x[lin+1][col]==0) {
      casa++;       }
    if (lin==1 &&x[2][col]==0 && x[3][col]==0){
      casa++;         }
    if (lin<7 && col<7 &&x[lin+1][col+1]<0) {
      casa++;         }
    if (lin<7 && col>0 &&x[lin+1][col-1]<0) {
      casa++;         }      }
  return  casa;     }
```

Torre

```
int torre(int x[8][8], int lin, int col){
  int casa = 0;
  int nl = lin-1;
  int nc =col;
  while (nl>=0){
    if (x[nl][nc]*x[lin][col]==0) {
      casa++;  }
    if (x[nl][nc]*x[lin][col]<0) {
      casa++; nl=0; }
    if (x[nl][nc]*x[lin][col]>0) {
      nl=0; }
      nl++; }
    nl=lin+1;
  // ... nl<= 7...
  // ... nc<= 7...
  // ... nc>= 7...
  return casa;}
```

Cavalo

```
int cavalo(int x[8][8], int lin, int col){
  int i,nl,nc,casa = 0;
  int dlin[8]={-2,-1,1,2,2,1,-1,-2};
  int dcol[8]={1,2,2,1,-1,-2,-2,-1};
  for (i=0;i<8;i++){
    nl=lin+dlin[i];
    nc=col+dcol[i];
    if (nl<=7 && nl>=0 && nc<=7 && nc>=0){
      if (x[nl][nc]==0||x[nl][nc]*x[lin][col]<0){
        casa++; }  }   }
  return casa;
```

Bispo

```
int bispo(int x[8][8], int lin, int col){
  int nl,nc,casa=0;
  nl=lin-1;
  nc=col-1;
  while (nl>=0 && nc>=0){
    if (x[nl][nc]==0) {
      casa++; }
    if (x[nl][nc]*x[lin][col]<0){
      casa++;   nl=0;}
    if (x[nl][nc]*x[lin][col]>0){
      nl=0; }
    nl--;
    nc--;  }
  nl=lin-1;
  nc=col+1;
  // while nl>=0 && nc<=7...
  // while nl<=7 &&nc >=0...
  // while nl<=7 && nc<=7...
  return casa;       }
```

Dama e Rei Fazendo os devidos ajustes aos controles de torre e bispo, é quase trivial. Ficam para você fazer, por falta de espaço.

☞ Para você fazer

A seguir um tabuleiro em situação de meio jogo, seguido do endereço de 3 peças. Escreva um programa capaz de descobrir quantas jogadas cada peça individualizada pode realizar na jogada a seguir.

```
     0   1   2   3   4   5   6   7
  +-------------------------------
0|   3  -3   0   0  -6   0   0   0
1|  -1   0  -1   0   0   0   0   0
2|   0   0   0   0  -1   0  -1  -1
3|   0  -1   0  -1   0   0   0  -4
4|  -4   0   0   0   0  -1   1   1
5|   0  -5   0   0   0   0   0   0
6|   0   1   1   1   1   1   0   0
7|   0   3   4   5   6   4   0   0
```

Descubra quantas jogadas podem ser feitas pelas peças localizadas em: 1) 1 0 , 2) 7 2 e 3) 4 0 .

Responda aqui:

1	2	3

Gerador de movimentos de xadrez

O xadrez tem um papel fundamental na Inteligência Artificial, e por conseguinte na ciência da computação. Ele é suficientemente compacto para poder ser manuseado dentro de um computador, mesmo um computador da década de 50, com poucas centenas de bytes de memória. Mas, é abrangente e complexo para representar o grau máximo de abstração e simbolismo de que o ser humano é capaz. Turing, o genial criador desta ciência, sugeriu em 1951 que *"em 50 anos, um computador será capaz de vencer o campeão mundial de xadrez"*. Ele errou por pouco: 1951+50=2001. Em 1997, Deep Blue, da IBM, venceu Gary Kasparov que era o campeão mundial. Que outra previsão – no mundo da computação – tem, teve ou terá tal grau de acerto ? Já foi chamado de *drosóphila melanogaster* da IA, similar à importância da mosca doméstica na genética.

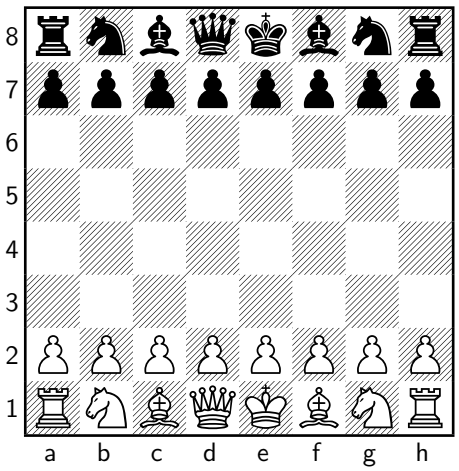
Abstraindo a computação, ainda assim o xadrez é um mundo maravilhoso. Aqui no Brasil, não temos grande tradição nele, mas em uma boa parte do mundo, as crianças aprendem a jogar e jogam durante toda sua vida escolar e por conseguinte muitos adultos seguem jogando vida afora. Isto, desde o século XII, que registra as primeiras manifestações do jogo, que por sinal é muito mais antigo do que isso.

Ele representa 2 exércitos que se opõe em um tabuleiro limitado (e pequeno) de 64 casas, e o único objetivo do jogo é vencer o exército inimigo. Esse fato é representado pela captura do rei oposto. Compõe o exército, as forças usuais: a infantaria, representada pelos peões em número de 8. As máquinas de guerra, as torres em número de 2. A igreja, personagem sempre influente, pelos seus 2 bispos. A cavalaria com 2 cavalos. Finalmente o casal real: a raínha ou dama, a peça mais poderosa no tabuleiro e o rei, limitado em movimentos, mas a peça em volta da qual todos operam.

Jogar xadrez é uma viagem intelectual indescritível, puro prazer. É superado pela construção de um programa de computador que jogue xadrez. Não é tarefa simples, mas deveria ser um investimento intelectual de todos quantos pretendam ser programadores plenos de computadores.

Como sempre, um programa grande e complexo precisa ser dividido em funções menores em tamanho e escopo. Uma parte importante em qualquer jogador é a análise e descoberta de todas as jogadas possíveis de uma determinada peça. Este exercício vai pedir que você programe uma pequena parte desta função.

Preliminares Antes de começar, é preciso se familiarizar com um tabuleiro armado.



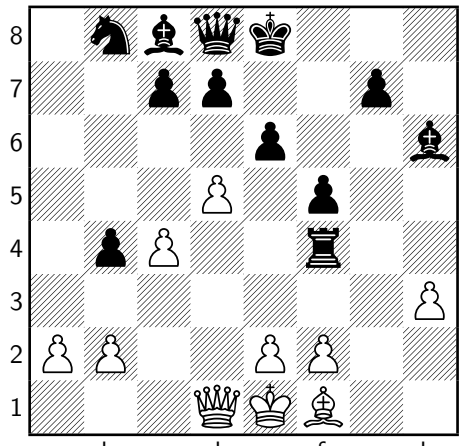
Tal disposição é boa para humanos, mas computadores precisam outra coisa: uma matriz numérica de 8 × 8, com casa brancas e negras alternadas sendo que que a casa embaixo, mais à direita, é branca. Por ser numérica, a matriz usa números para identificar as peças. Primeiro, a cor. Peças brancas são positivas e negras, negativas. Os valores: 1=peão, 2=torre, 3=cavalo, 4=bispo, 5=dama e 6=rei. Note que a célula linha 0, coluna 0, é a superior esquerda da matriz. Nesses termos, o tabuleiro acima corresponde a

	0	1	2	3	4	5	6	7
0|	-2	-3	-4	-5	-6	-4	-3	-2
1|	-1	-1	-1	-1	-1	-1	-1	-1
2|	0	0	0	0	0	0	0	0
3|	0	0	0	0	0	0	0	0
4|	0	0	0	0	0	0	0	0
5|	0	0	0	0	0	0	0	0
6|	1	1	1	1	1	1	1	1
7|	2	3	4	5	6	4	3	2

Você deve escrever um programa que vai receber um tabuleiro preenchido com uma situação de meio-jogo e também uma linha (0-7) e coluna (0-7). Você deve olhar o tabuleiro fornecido e ver qual é a peça que está lá. Identificada a peça e a sua cor, deve descobrir quantas casas essa peça pode ocupar num eventual próximo lance. Lembrando que as casas examinadas:

- se contiverem zero, podem ser ocupadas.
- se contiverem valor de sinal oposto ao da peça podem ser ocupadas, mas este valor implica parar na busca nessa direção (exceto o cavalo).
- se contiverem valor de mesmo sinal, não podem ser ocupadas e determinam final da busca.
- A casa não deve ser considerada se algum índice (linha ou coluna ou ambos) cair fora do tabuleiro.

Finalmente, não se assuste se aparecer um tabuleiro com alguma situação muito infrequente do ponto de vista enxadrístico: o gerador usa e abusa de números aleatórios. Vejamos um exemplo



Se o endereço a pesquisar fosse 4,5 a resposta deveria ser 7. Se fosse 1,2 a resposta deveria ser 2. Se fosse 0,1 a resposta deveria ser 2.

Movimento do peão Se na casa inicial, pode adiantar 1 ou 2 casas. Se em outras casas, só pode andar 1 posição. Pode tomar em diagonal, à direita ou à esquerda. Não se implementa aqui a tomada *en passant*, pois ela depende da jogada anterior, que neste caso é desconhecida.

```
int peao(int x[8][8], int lin, int col){
  int casa = 0;
  if (x[lin][col]==1) {
    if (lin > 0 && x[lin-1][col]==0) {
      casa++;      }
    if (lin==6 &&x[4][col]==0 && x[5][col]==0){
      casa++;     }
    if (lin>0 && col<7 &&x[lin-1][col+1]<0) {
      casa++;         }
    if (lin>0 && col>0 &&x[lin-1][col-1]<0) {
      casa++;          }       }
  else {
    if (lin <7 && x[lin+1][col]==0) {
      casa++;       }
    if (lin==1 &&x[2][col]==0 && x[3][col]==0){
      casa++;         }
    if (lin<7 && col<7 &&x[lin+1][col+1]<0) {
      casa++;         }
    if (lin<7 && col>0 &&x[lin+1][col-1]<0) {
      casa++;         }      }
  return  casa;     }
```

Torre

```
int torre(int x[8][8], int lin, int col){
  int casa = 0;
  int nl = lin-1;
  int nc =col;
  while (nl>=0){
    if (x[nl][nc]*x[lin][col]==0) {
      casa++;  }
    if (x[nl][nc]*x[lin][col]<0) {
      casa++; nl=0; }
    if (x[nl][nc]*x[lin][col]>0) {
      nl=0; }
      nl++; }
    nl=lin+1;
  // ... nl<= 7...
  // ... nc<= 7...
  // ... nc>= 7...
  return casa;}
```

Cavalo

```
int cavalo(int x[8][8], int lin, int col){
  int i,nl,nc,casa = 0;
  int dlin[8]={-2,-1,1,2,2,1,-1,-2};
  int dcol[8]={1,2,2,1,-1,-2,-2,-1};
  for (i=0;i<8;i++){
    nl=lin+dlin[i];
    nc=col+dcol[i];
    if (nl<=7 && nl>=0 && nc<=7 && nc>=0){
      if (x[nl][nc]==0||x[nl][nc]*x[lin][col]<0){
        casa++; }  }   }
  return casa;
```

Bispo

```
int bispo(int x[8][8], int lin, int col){
  int nl,nc,casa=0;
  nl=lin-1;
  nc=col-1;
  while (nl>=0 && nc>=0){
    if (x[nl][nc]==0) {
      casa++; }
    if (x[nl][nc]*x[lin][col]<0){
      casa++;   nl=0;}
    if (x[nl][nc]*x[lin][col]>0){
      nl=0; }
    nl--;
    nc--;  }
  nl=lin-1;
  nc=col+1;
  // while nl>=0 && nc<=7...
  // while nl<=7 &&nc >=0...
  // while nl<=7 && nc<=7...
  return casa;       }
```

Dama e Rei Fazendo os devidos ajustes aos controles de torre e bispo, é quase trivial. Ficam para você fazer, por falta de espaço.

☞ Para você fazer

A seguir um tabuleiro em situação de meio jogo, seguido do endereço de 3 peças. Escreva um programa capaz de descobrir quantas jogadas cada peça individualizada pode realizar na jogada a seguir.

	0	1	2	3	4	5	6	7
0|	0	-3	0	0	-6	0	0	0
1|	-1	0	0	-1	0	0	0	4
2|	-2	-1	-1	0	-5	-1	-1	0
3|	0	5	0	0	0	0	0	0
4|	0	0	0	0	0	-3	0	0
5|	2	0	1	0	1	0	0	0
6|	0	0	3	1	0	0	0	0
7|	0	3	2	0	6	0	0	0

Descubra quantas jogadas podem ser feitas pelas peças localizadas em: 1) 2 5 , 2) 1 7 e 3) 5 0 .

Responda aqui:

1	2	3

Gerador de movimentos de xadrez

O xadrez tem um papel fundamental na Inteligência Artificial, e por conseguinte na ciência da computação. Ele é suficientemente compacto para poder ser manuseado dentro de um computador, mesmo um computador da década de 50, com poucas centenas de bytes de memória. Mas, é abrangente e complexo para representar o grau máximo de abstração e simbolismo de que o ser humano é capaz. Turing, o genial criador desta ciência, sugeriu em 1951 que *"em 50 anos, um computador será capaz de vencer o campeão mundial de xadrez"*. Ele errou por pouco: 1951+50=2001. Em 1997, Deep Blue, da IBM, venceu Gary Kasparov que era o campeão mundial. Que outra previsão – no mundo da computação – tem, teve ou terá tal grau de acerto ? Já foi chamado de *drosóphila melanogaster* da IA, similar à importância da mosca doméstica na genética.

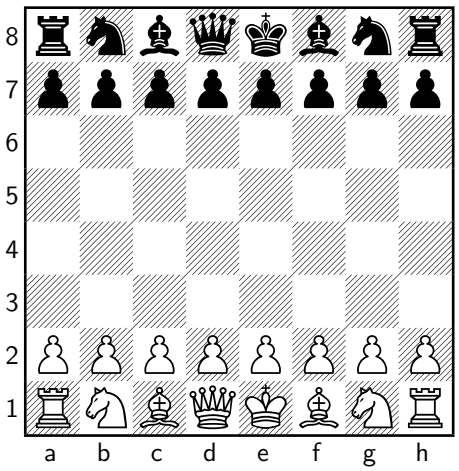
Abstraindo a computação, ainda assim o xadrez é um mundo maravilhoso. Aqui no Brasil, não temos grande tradição nele, mas em uma boa parte do mundo, as crianças aprendem a jogar e jogam durante toda sua vida escolar e por conseguinte muitos adultos seguem jogando vida afora. Isto, desde o século XII, que registra as primeiras manifestações do jogo, que por sinal é muito mais antigo do que isso.

Ele representa 2 exércitos que se opõe em um tabuleiro limitado (e pequeno) de 64 casas, e o único objetivo do jogo é vencer o exército inimigo. Esse fato é representado pela captura do rei oposto. Compõe o exército, as forças usuais: a infantaria, representada pelos peões em número de 8. As máquinas de guerra, as torres em número de 2. A igreja, personagem sempre influente, pelos seus 2 bispos. A cavalaria com 2 cavalos. Finalmente o casal real: a raínha ou dama, a peça mais poderosa no tabuleiro e o rei, limitado em movimentos, mas a peça em volta da qual todos operam.

Jogar xadrez é uma viagem intelectual indescritível, puro prazer. É superado pela construção de um programa de computador que jogue xadrez. Não é tarefa simples, mas deveria ser um investimento intelectual de todos quantos pretendam ser programadores plenos de computadores.

Como sempre, um programa grande e complexo precisa ser dividido em funções menores em tamanho e escopo. Uma parte importante em qualquer jogador é a análise e descoberta de todas as jogadas possíveis de uma determinada peça. Este exercício vai pedir que você programe uma pequena parte desta função.

Preliminares Antes de começar, é preciso se familiarizar com um tabuleiro armado.



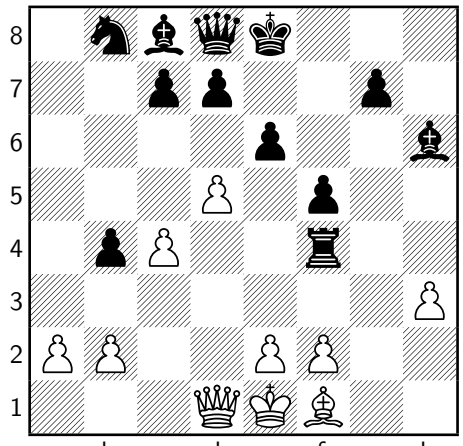
Tal disposição é boa para humanos, mas computadores precisam outra coisa: uma matriz numérica de 8 × 8, com casa brancas e negras alternadas sendo que que a casa embaixo, mais à direita, é branca. Por ser numérica, a matriz usa números para identificar as peças. Primeiro, a cor. Peças brancas são positivas e negras, negativas. Os valores: 1=peão, 2=torre, 3=cavalo, 4=bispo, 5=dama e 6=rei. Note que a célula linha 0, coluna 0, é a superior esquerda da matriz. Nesses termos, o tabuleiro acima corresponde a

	0	1	2	3	4	5	6	7
0|	-2	-3	-4	-5	-6	-4	-3	-2
1|	-1	-1	-1	-1	-1	-1	-1	-1
2|	0	0	0	0	0	0	0	0
3|	0	0	0	0	0	0	0	0
4|	0	0	0	0	0	0	0	0
5|	0	0	0	0	0	0	0	0
6|	1	1	1	1	1	1	1	1
7|	2	3	4	5	6	4	3	2

Você deve escrever um programa que vai receber um tabuleiro preenchido com uma situação de meio-jogo e também uma linha (0-7) e coluna (0-7). Você deve olhar o tabuleiro fornecido e ver qual é a peça que está lá. Identificada a peça e a sua cor, deve descobrir quantas casas essa peça pode ocupar num eventual próximo lance. Lembrando que as casas examinadas:

- se contiverem zero, podem ser ocupadas.
- se contiverem valor de sinal oposto ao da peça podem ser ocupadas, mas este valor implica parar na busca nessa direção (exceto o cavalo).
- se contiverem valor de mesmo sinal, não podem ser ocupadas e determinam final da busca.
- A casa não deve ser considerada se algum índice (linha ou coluna ou ambos) cair fora do tabuleiro.

Finalmente, não se assuste se aparecer um tabuleiro com alguma situação muito infrequente do ponto de vista enxadrístico: o gerador usa e abusa de números aleatórios. Vejamos um exemplo



Se o endereço a pesquisar fosse 4,5 a resposta deveria ser 7. Se fosse 1,2 a resposta deveria ser 2. Se fosse 0,1 a resposta deveria ser 2.

Movimento do peão Se na casa inicial, pode adiantar 1 ou 2 casas. Se em outras casas, só pode andar 1 posição. Pode tomar em diagonal, à direita ou à esquerda. Não se implementa aqui a tomada *en passant*, pois ela depende da jogada anterior, que neste caso é desconhecida.

```
int peao(int x[8][8], int lin, int col){
  int casa = 0;
  if (x[lin][col]==1) {
    if (lin > 0 && x[lin-1][col]==0) {
      casa++;      }
    if (lin==6 &&x[4][col]==0 && x[5][col]==0){
      casa++;     }
    if (lin>0 && col<7 &&x[lin-1][col+1]<0) {
      casa++;         }
    if (lin>0 && col>0 &&x[lin-1][col-1]<0) {
      casa++;          }       }
  else {
    if (lin <7 && x[lin+1][col]==0) {
      casa++;       }
    if (lin==1 &&x[2][col]==0 && x[3][col]==0){
      casa++;         }
    if (lin<7 && col<7 &&x[lin+1][col+1]<0) {
      casa++;         }
    if (lin<7 && col>0 &&x[lin+1][col-1]<0) {
      casa++;         }      }
  return  casa;     }
```

Torre

```
int torre(int x[8][8], int lin, int col){
  int casa = 0;
  int nl = lin-1;
  int nc =col;
  while (nl>=0){
    if (x[nl][nc]*x[lin][col]==0) {
      casa++;  }
    if (x[nl][nc]*x[lin][col]<0) {
      casa++; nl=0; }
    if (x[nl][nc]*x[lin][col]>0) {
      nl=0; }
      nl++; }
    nl=lin+1;
  // ... nl<= 7...
  // ... nc<= 7...
  // ... nc>= 7...
  return casa;}
```

Cavalo

```
int cavalo(int x[8][8], int lin, int col){
  int i,nl,nc,casa = 0;
  int dlin[8]={-2,-1,1,2,2,1,-1,-2};
  int dcol[8]={1,2,2,1,-1,-2,-2,-1};
  for (i=0;i<8;i++){
    nl=lin+dlin[i];
    nc=col+dcol[i];
    if (nl<=7 && nl>=0 && nc<=7 && nc>=0){
      if (x[nl][nc]==0||x[nl][nc]*x[lin][col]<0){
        casa++; }  }   }
  return casa;
```

Bispo

```
int bispo(int x[8][8], int lin, int col){
  int nl,nc,casa=0;
  nl=lin-1;
  nc=col-1;
  while (nl>=0 && nc>=0){
    if (x[nl][nc]==0) {
      casa++; }
    if (x[nl][nc]*x[lin][col]<0){
      casa++;   nl=0;}
    if (x[nl][nc]*x[lin][col]>0){
      nl=0; }
    nl--;
    nc--;  }
  nl=lin-1;
  nc=col+1;
  // while nl>=0 && nc<=7...
  // while nl<=7 &&nc >=0...
  // while nl<=7 && nc<=7...
  return casa;       }
```

Dama e Rei Fazendo os devidos ajustes aos controles de torre e bispo, é quase trivial. Ficam para você fazer, por falta de espaço.

☞ Para você fazer

A seguir um tabuleiro em situação de meio jogo, seguido do endereço de 3 peças. Escreva um programa capaz de descobrir quantas jogadas cada peça individualizada pode realizar na jogada a seguir.

	0	1	2	3	4	5	6	7
0|	0	0	0	4	-6	-3	-3	0
1|	0	-1	0	5	-1	-4	-1	0
2|	-1	0	0	0	0	-1	0	0
3|	-2	0	0	0	0	2	0	0
4|	0	1	0	-1	0	0	0	0
5|	0	0	0	0	0	0	1	1
6|	1	0	1	1	0	0	0	0
7|	0	3	0	0	6	4	0	2

Descubra quantas jogadas podem ser feitas pelas peças localizadas em: 1) 6 2 , 2) 7 7 e 3) 0 5 .

Responda aqui:

1	2	3

UFPR - UP - UTFPR - PUCPr – 27/06/2022 - 11:31:05.7
Prof Dr P Kantek (pkantek@gmail.com)
Gerador de movimentos de xadrez VIVO050c, V: 1.06 5
- 5 _____ / _____ / _____

Gerador de movimentos de xadrez

O xadrez tem um papel fundamental na Inteligência Artificial, e por conseguinte na ciência da computação. Ele é suficientemente compacto para poder ser manuseado dentro de um computador, mesmo um computador da década de 50, com poucas centenas de bytes de memória. Mas, é abrangente e complexo para representar o grau máximo de abstração e simbolismo de que o ser humano é capaz. Turing, o genial criador desta ciência, sugeriu em 1951 que *"em 50 anos, um computador será capaz de vencer o campeão mundial de xadrez"*. Ele errou por pouco: 1951+50=2001. Em 1997, Deep Blue, da IBM, venceu Gary Kasparov que era o campeão mundial. Que outra previsão – no mundo da computação – tem, teve ou terá tal grau de acerto ? Já foi chamado de *drosóphila melanogaster* da IA, similar à importância da mosca doméstica na genética.

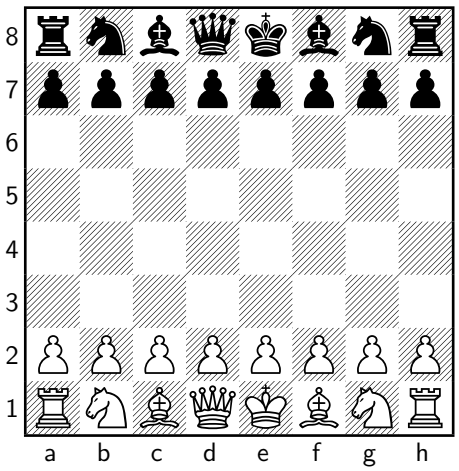
Abstraindo a computação, ainda assim o xadrez é um mundo maravilhoso. Aqui no Brasil, não temos grande tradição nele, mas em uma boa parte do mundo, as crianças aprendem a jogar e jogam durante toda sua vida escolar e por conseguinte muitos adultos seguem jogando vida afora. Isto, desde o século XII, que registra as primeiras manifestações do jogo, que por sinal é muito mais antigo do que isso.

Ele representa 2 exércitos que se opõe em um tabuleiro limitado (e pequeno) de 64 casas, e o único objetivo do jogo é vencer o exército inimigo. Esse fato é representado pela captura do rei oposto. Compõe o exército, as forças usuais: a infantaria, representada pelos peões em número de 8. As máquinas de guerra, as torres em número de 2. A igreja, personagem sempre influente, pelos seus 2 bispos. A cavalaria com 2 cavalos. Finalmente o casal real: a raínha ou dama, a peça mais poderosa no tabuleiro e o rei, limitado em movimentos, mas a peça em volta da qual todos operam.

Jogar xadrez é uma viagem intelectual indescritível, puro prazer. É superado pela construção de um programa de computador que jogue xadrez. Não é tarefa simples, mas deveria ser um investimento intelectual de todos quantos pretendam ser programadores plenos de computadores.

Como sempre, um programa grande e complexo precisa ser dividido em funções menores em tamanho e escopo. Uma parte importante em qualquer jogador é a análise e descoberta de todas as jogadas possíveis de uma determinada peça. Este exercício vai pedir que você programe uma pequena parte desta função.

Preliminares Antes de começar, é preciso se familiarizar com um tabuleiro armado.



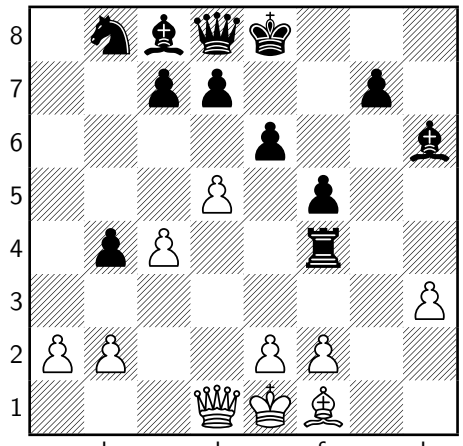
Tal disposição é boa para humanos, mas computadores precisam outra coisa: uma matriz numérica de 8 × 8, com casa brancas e negras alternadas sendo que que a casa embaixo, mais à direita, é branca. Por ser numérica, a matriz usa números para identificar as peças. Primeiro, a cor. Peças brancas são positivas e negras, negativas. Os valores: 1=peão, 2=torre, 3=cavalo, 4=bispo, 5=dama e 6=rei. Note que a célula linha 0, coluna 0, é a superior esquerda da matriz. Nesses termos, o tabuleiro acima corresponde a

```
     0   1   2   3   4   5   6   7
  +-------------------------------
0|  -2  -3  -4  -5  -6  -4  -3  -2
1|  -1  -1  -1  -1  -1  -1  -1  -1
2|   0   0   0   0   0   0   0   0
3|   0   0   0   0   0   0   0   0
4|   0   0   0   0   0   0   0   0
5|   0   0   0   0   0   0   0   0
6|   1   1   1   1   1   1   1   1
7|   2   3   4   5   6   4   3   2
```

Você deve escrever um programa que vai receber um tabuleiro preenchido com uma situação de meio-jogo e também uma linha (0-7) e coluna (0-7). Você deve olhar o tabuleiro fornecido e ver qual é a peça que está lá. Identificada a peça e a sua cor, deve descobrir quantas casas essa peça pode ocupar num eventual próximo lance. Lembrando que as casas examinadas:

- se contiverem zero, podem ser ocupadas.
- se contiverem valor de sinal oposto ao da peça podem ser ocupadas, mas este valor implica parar na busca nessa direção (exceto o cavalo).
- se contiverem valor de mesmo sinal, não podem ser ocupadas e determinam final da busca.
- A casa não deve ser considerada se algum índice (linha ou coluna ou ambos) cair fora do tabuleiro.

Finalmente, não se assuste se aparecer um tabuleiro com alguma situação muito infrequente do ponto de vista enxadrístico: o gerador usa e abusa de números aleatórios. Vejamos um exemplo



Se o endereço a pesquisar fosse 4,5 a resposta deveria ser 7. Se fosse 1,2 a resposta deveria ser 2. Se fosse 0,1 a resposta deveria ser 2.

Movimento do peão Se na casa inicial, pode adiantar 1 ou 2 casas. Se em outras casas, só pode andar 1 posição. Pode tomar em diagonal, à direita ou à esquerda. Não se implementa aqui a tomada *en passant*, pois ela depende da jogada anterior, que neste caso é desconhecida.

```
int peao(int x[8][8], int lin, int col){
  int casa = 0;
  if (x[lin][col]==1) {
    if (lin > 0 && x[lin-1][col]==0) {
      casa++;      }
    if (lin==6 &&x[4][col]==0 && x[5][col]==0){
      casa++;     }
    if (lin>0 && col<7 &&x[lin-1][col+1]<0) {
      casa++;         }
    if (lin>0 && col>0 &&x[lin-1][col-1]<0) {
      casa++;          }       }
  else {
    if (lin <7 && x[lin+1][col]==0) {
      casa++;       }
    if (lin==1 &&x[2][col]==0 && x[3][col]==0){
      casa++;         }
    if (lin<7 && col<7 &&x[lin+1][col+1]<0) {
      casa++;         }
    if (lin<7 && col>0 &&x[lin+1][col-1]<0) {
      casa++;         }      }
  return  casa;     }
```

Torre

```
int torre(int x[8][8], int lin, int col){
  int casa = 0;
  int nl = lin-1;
  int nc =col;
  while (nl>=0){
    if (x[nl][nc]*x[lin][col]==0) {
      casa++;  }
    if (x[nl][nc]*x[lin][col]<0) {
      casa++; nl=0; }
    if (x[nl][nc]*x[lin][col]>0) {
      nl=0; }
      nl++; }
    nl=lin+1;
  // ... nl<= 7...
  // ... nc<= 7...
  // ... nc>= 7...
  return casa;}
```

Cavalo

```
int cavalo(int x[8][8], int lin, int col){
  int i,nl,nc,casa = 0;
  int dlin[8]={-2,-1,1,2,2,1,-1,-2};
  int dcol[8]={1,2,2,1,-1,-2,-2,-1};
  for (i=0;i<8;i++){
    nl=lin+dlin[i];
    nc=col+dcol[i];
    if (nl<=7 && nl>=0 && nc<=7 && nc>=0){
      if (x[nl][nc]==0||x[nl][nc]*x[lin][col]<0){
        casa++; }  }   }
  return casa;
```

Bispo

```
int bispo(int x[8][8], int lin, int col){
  int nl,nc,casa=0;
  nl=lin-1;
  nc=col-1;
  while (nl>=0 && nc>=0){
    if (x[nl][nc]==0) {
      casa++; }
    if (x[nl][nc]*x[lin][col]<0){
      casa++;   nl=0;}
    if (x[nl][nc]*x[lin][col]>0){
      nl=0; }
    nl--;
    nc--;  }
  nl=lin-1;
  nc=col+1;
  // while nl>=0 && nc<=7...
  // while nl<=7 &&nc >=0...
  // while nl<=7 && nc<=7...
  return casa;       }
```

Dama e Rei Fazendo os devidos ajustes aos controles de torre e bispo, é quase trivial. Ficam para você fazer, por falta de espaço.

☞ Para você fazer

A seguir um tabuleiro em situação de meio jogo, seguido do endereço de 3 peças. Escreva um programa capaz de descobrir quantas jogadas cada peça individualizada pode realizar na jogada a seguir.

```
     0   1   2   3   4   5   6   7
  +-------------------------------
0|   0   0  -5   0  -6  -4  -3   0
1|   0   0  -1   0   0   0  -2   0
2|   0   0   0   0   0   0   0  -1
3|   0   0   1  -1  -1   0   0   0
4|   1   0   2   0   0   0   0   1
5|   0   0  -2   0   0   0   1   0
6|   0   1   0   0   0   5   0   0
7|   0   3   4   0   6   4   0   0
```

Descubra quantas jogadas podem ser feitas pelas peças localizadas em: 1) 1 6 , 2) 2 7 e 3) 0 5 .

Responda aqui:

1	2	3

```
===== 27/06/2022 11:31:05.7 =====E=PL050c
    1         7    5    0
    2         2    0    0
    3         1    2    6
    4         2    2    4
    5         8    1    3
```