MAC 115 – Introdução à Ciência da Computação

Terceiro Exercício-Programa

Busca minas

Muitos de vocês devem conhecer o jogo do campo minado. Tem implementações para o Windows –*Minesweeper* ou *Campo Minado* – e para o Linux – *KMines* e *Gnomine*. O objetivo desse exercício–programa é implementar uma versão deste jogo. Como ainda estamos começando, não vai ter a parte gráfica, mas vamos quebrar o galho.

Descrição do jogo

Inicialmente o usuário define o tamanho do tabuleiro do jogo ($m \times n$) e o número de minas z que devem estar escondidas lá dentro. A seguir as z minas são distribuídas aleatoriamente no tabuleiro.

No início do jogo todas as posições do tabuleiro estão fechadas. A partir daí o usuário "chuta" posições do tabuleiro do jogo onde ele espera que não existam minas. Se ele chutar uma posição com uma mina o jogo acaba (afinal o usuário morreu...), senão o tabuleiro é novamente mostrado na tela com essa posição aberta, mostrando o número de minas vizinhas a essa posição. Caso não hajam minas vizinhas à posição que o usuário forneceu, então todos os vizinhos são abertos. Observe que pode ocorrer que uma dessas posições abertas seja novamente uma posição que não tem minas vizinhas, e então todos os seus vizinhos devem ser abertos e assim sucessivamente.

O objetivo do jogo é abrir todas as $m \times n - z$ posições do tabuleiro que não têm minas.

Exemplo: Dados $m = 4$, $n = 4$ e $z = 5$. Suponhamos que as posições com minas no tabuleiro sejam (1,1), (2,1), (3,1), (1,4) e (2,4) (claro que o usuário não sabe disso!). As coordenadas têm a forma (linha, coluna);

```
De o numero de linhas do tabuleiro do jogo: 4
De o numero de colunas do tabuleiro do jogo: 4
De o numero de minas: 5
De a semente: 327
```

```
Numero de linhas (3 <= m <= 90): 4
Numero de colunas (3 <= n <= 90): 4
Numero de minas (1 <= z <= 16): 5
Semente (1 <= s <= 12345): 327
Bem vindo ao BuscaMinas!
       1  2  3  4
    +-------------+
  1 |  .  .  .  . | 1
  2 |  .  .  .  . | 2
  3 |  .  .  .  . | 3
  4 |  .  .  .  . | 4
    +-------------+
       1  2  3  4

Proximo chute: a 4 1
Por enquanto: 0/5 marcadas, 15 livres.
       1  2  3  4
    +-------------+
  1 |  .  .  .  . | 1
  2 |  .  .  .  . | 2
  3 |  .  .  .  . | 3
  4 |  1  .  .  . | 4
    +-------------+
       1  2  3  4

Proximo chute: a 4 4
Por enquanto: 0/5 marcadas, 9 livres.
       1  2  3  4
    +-------------+
  1 |  .  .  .  . | 1
  2 |  .  .  .  . | 2
  3 |  .  2  1  1 | 3
  4 |  1  1  0  0 | 4
    +-------------+
       1  2  3  4

Proximo chute: a 3 3
Sem efeito.

Proximo chute: m 4 2
Sem efeito.
```

```
Proximo chute: m 2 2
Por enquanto: 1/5 marcadas, 8 livres.
       1  2  3  4
    +-------------+
  1 |  .  .  .  . | 1
  2 |  .  *  .  . | 2
  3 |  .  2  1  1 | 3
  4 |  1  1  0  0 | 4
    +-------------+
       1  2  3  4

Proximo chute: a 1 1
BOOOOM!  Voce acaba de pisar numa mina!
       1  2  3  4
    +-------------+
  1 |  @  2  2  @ | 1
  2 |  @  3  2  @ | 2
  3 |  @  2  1  1 | 3
  4 |  1  1  0  0 | 4
    +-------------+
       1  2  3  4
```

Seu programa deve simular a execução desse jogo. Para isso, o programa deve ler os valores m, n e z dados pelo usuário e criar um tabuleiro de jogo T com

$$T(i,j) = \begin{cases} -1, & \text{caso haja uma mina na posição } (i,j), \\ k, & \text{caso contrário,} \end{cases}$$

para $1 \leq i \leq m$ e $1 \leq j \leq n$, onde k é o número de minas nas posições adjacentes a (i, j) no tabuleiro (no máximo, 8). Declare no seu programa valores máximos `MMAX=90` e `NMAX=90` e verifique se os valores inseridos pelo usuário respeitam esses limites. Seu programa deve funcionar para qualquer valor de n entre 3 e `NMAX` e qualquer valor de m entre 3 e `MMAX`. Se os valores de m, n ou z digitados pelo jogador forem incorretos, novos valores devem ser solicitados (até que valores corretos sejam digitados). Note que valores de m e n fora dos intervalos especificados são incorretos assim como $z \geq m \times n$.

A seguir, seu programa deverá ler uma sequência de comandos fornecidos pelo usuário, cada um da forma

letra linha coluna

onde (linha, coluna) deve ser uma coordenada válida, dentro do tabuleiro. A letra pode ser `A` (abrir), `M` (marcar) ou `D` (desmarcar), conforme explicado adiante (as letras minúsculas `amd` também devem ser aceitas, de forma equivalente). O jogo termina quando todas as posições livres estiverem abertas ou se o jogador tenta abrir uma posição com uma mina. Para evitar que o jogador perca na primeira jogada, sorteie as posições com minas após ter lido o primeiro comando evitando colocar uma mina na posição correspondente.

O comando para abrir a posição (i, j) é `A i j`. A primeira jogada do jogo deve obrigatoriamente ser uma jogada deste tipo, assim, se receber um `M` ou `D` na primeira jogada, trate como se fosse `A`. Quando o jogador faz uma jogada do tipo `A i j`, caso a posição (i, j) esteja ocupada por uma mina, você deve dar uma mensagem, imprimir o tabuleiro com todas as posições abertas e o jogo acaba. Se a posição fornecida já estiver aberta, imprima uma mensagem e continue o jogo. Caso seja uma posição fechada e que não esteja ocupada por uma mina, abra essa posição e, caso seja uma posição sem minas vizinhas, seus vizinhos (e caso um desses vizinhos também não tenha minas vizinhas, seus vizinhos, e assim por diante - em outras palavras, toda posição que seja vizinha de uma aberta e sem minas vizinhas deve ser aberta também).

Para lembrar onde você acha que tem uma mina, é possível *marcar* a posição. O comando `M i j`, se a posição (i, j) estiver fechada, faz com que ela seja marcada com um `*` e o tabuleiro exibido novamente. Se a posição (i, j) já estiver aberta, uma mensagem deve ser exibida e o jogo deve continuar. **O número de casas marcadas não pode ultrapassar o número de minas; tentar marcar uma casa além disso não deve alterar nada.**

O jogador deve também ter a chance de desmarcar uma posição marcada. Para isso, ele poderá digitar o comando `D i j`. Se a posição (i, j) estiver marcada, ela deve ser desmarcada e o tabuleiro exibido novamente. Se a posição (i, j) não estiver marcada ou já estiver aberta, deve ser exibida uma mensagem e o jogo deve continuar.

Note que é importante saber, para cada posição do tabuleiro, se ela está aberta, fechada ou fechada e marcada, assim vale a pena ter uma matriz registrando isso.

Para sortear as posições das minas no tabuleiro você deve usar a função de geração de números aleatórios do C. Na biblioteca `stdlib.h` existem as funções `rand()` e `srand()`. A função `rand()` é do tipo `int`, sem parâmetro, e devolve o próximo número aleatório (um número inteiro entre 0 e RAND_MAX); a função `srand()` é do tipo `void`, recebe como parâmetro um número inteiro positivo ("seed", ou seja,"semente"), e inicializa o gerador com esse inteiro. Exemplos de chamada:

```
srand (25);       /*  Inicializa o gerador com a semente 25.   */

numero = rand();  /*  Gera o próximo número aleatório. */
```

O programa deve chamar `srand` uma única vez, logo no começo, usando a semente dada pelo jogador; daí para a frente só chama `rand`. A semente é pedida para o jogador para que o mesmo tabuleiro possa ser gerado em partidas diferentes, o que facilita encontrar e corrigir erros do programa. Um programa "de produção" procura semente de outra forma.

Quanto ao número aleatório devolvido por `rand`, ele é um inteiro grande, e pode-se extrair um par (i, j) da seguinte forma:

```
i = numero % m + 1;
j = (numero / m) % n + 1;
```

IMPORTANTE: Todo exercício-programa deve seguir as observações dadas em aula sobre as diretrizes para a forma de entrega do exercício, aspectos importantes na avaliação, etc.