MAC 110 – Introdução à Computação

Terceiro Exercício–Programa

O caçador de minas

Muitos de vocês devem conhecer um jogo bastante interessante do Windows: o *campo minado*. O objetivo desse exercício–programa é implementar uma versão deste jogo.

Descrição do jogo

Inicialmente o usuário define o tamanho da matriz 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 na matriz.

No início do jogo todas as posições da matriz estão fechadas. A partir daí o usuário "chuta" posições da matriz do jogo onde ele espera que não existam minas. Se ele chutar uma posição com mina o jogo acaba (afinal o usuário morreu...), senão a matriz é novamente mostrada 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 descobrir a posição de todas as minas, ou seja, abrir as $m.n - z$ posições da matriz que não têm minas.

Exemplo: Dado $m = 4$, $n = 4$ e $z = 5$. Suponhamos que as posições com minas na matriz sejam (1,1), (1,2), (1,3), (2,1) e (4,2) (claro que o usuário não sabe disso!!).

```
De o numero de linhas da matriz do jogo: 4
De o numero de colunas da matriz do jogo: 4
De o numero de minas: 5
```

```
Bem vindo ao caca minas!
Situacao atual (restam 11 posicoes livres)
      1 2 3 4
    +---------+
  1 | . . . .|  1
  2 | . . . .|  2
  3 | . . . .|  3
  4 | . . . .|  4
    +---------+
      1 2 3 4
Proximo chute:  2 4
Situacao atual (restam 10 posicoes livres)
      1 2 3 4
    +---------+
  1 | . . . .|  1
  2 | . . . 1|  2
  3 | . . . .|  3
  4 | . . . .|  4
    +---------+
      1 2 3 4
Proximo chute: 4 4
Situacao atual (restam 5 posicoes livres)
      1 2 3 4
    +---------+
  1 | . . . .|  1
  2 | . . 2 1|  2
  3 | . . 1 0|  3
  4 | . . 1 0|  4
    +---------+
      1 2 3 4
Proximo chute: 1 1
BOOOOOOOM!!! Voce acaba de pisar em uma mina
      1 2 3 4
    +---------+
  1 |-1-1-1 1|  1
  2 |-1 4 2 1|  2
  3 | 2 2 1 0|  3
  4 | 1-1 1 0|  4
    +---------+
      1 2 3 4
```

Seu programa deve simular a execução desse jogo, ou seja, seu programa deve ler os valores m, n e z dados pelo usuário e criar uma matriz A de jogo onde

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

onde k é o número de minas nas posições adjacentes a (i,j) na matriz (são, no máximo, 8). Além disso, seu programa deve manter um marcador para cada posição, indicando se ela está aberta ou fechada. Para sortear as posições das minas na matriz você pode usar a função de geração de números aleatórios do C (veja as funções `rand()` e `srand()` da biblioteca `stdlib.h`).

A seguir, seu programa deverá ler uma seqüência de posições da matriz fornecidas pelo usuário. Verifique inicialmente se é uma posição válida para as dimensões do tabuleiro. Caso a posição esteja ocupada por uma mina, você deve dar uma mensagem, imprimir o tabuleiro aberto 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). O jogo termina quando todas as posições livres estiverem abertas.