Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
SCC0222 – Laboratório de Introdução à Ciência da Computação I

Recuperação: Campo Minado

Professor: Dr. Rodrigo Fernandes de Mello (`mello@icmc.usp.br`)
Estagiário PAE: Fábio Henrique Gomes Sikansi (`fhenrique@usp.br`)
Martha Dais Ferreira (`daismf@icmc.usp.br`)
Monitor: Lucas Parras (`parraslucas@gmail.com`)
Loys Gibertoni (`loys.gibertoni@usp.br`)
Colaborador: Felipe Simões Lage Gomes Duarte (`fgduarte@icmc.usp.br`)

1 Objetivo do Trabalho

Você deverá implementar os diferentes estágios do clássico jogo "Campo Minado"

2 Campo Minado

Campo minado é um popular jogo de computador para um jogador. Foi inventado por Robert Donner em 1989 e tem como objectivo revelar um campo de minas sem que alguma seja detonada. Este jogo tem sido reescrito para as mais diversas plataformas, sendo a sua versão mais popular a que vem nativamente com o Microsoft Windows.

2.1 Regras

A área de jogo consiste num campo de quadrados retangular. Cada quadrado pode ser revelado clicando sobre ele, e se o quadrado clicado contiver uma mina, então o jogo acaba. Se, por outro lado, o quadrado não contiver uma mina, uma de duas coisas poderá acontecer:

1. Um número aparece, indicando a quantidade de quadrados adjacentes que contêm minas;
2. Nenhum número aparece. Neste caso, o jogo revela automaticamente os quadrados que se encontram adjacentes ao quadrado vazio, já que não podem conter minas;

O jogo é ganho quando todos os quadrados que não têm minas são revelados.

3 Proposta

Seu programa deverá resolver as diversas etapas do jogo campo minado. Para facilitar o processo de desenvolvimento e correção, seu programa deverá implementar 3 diferentes módulos: (i) Leitura, (ii) inicialização do tabuleiro e (iii) ação do usuário. Cada um destes módulos deverão ser executados de acordo com uma opção inteira que será fornecida via `stdin`. Em seguida, ele deverá ler o nome do arquivo que contém o tabuleiro inicial do jogo. Deste modo, um exemplo de arquivo de entrada é:

```
1
1.board
```

3.1 Tabuleiro

O arquivo que contém o tabuleiro é formado por caracteres `'.'` que indicam espaço vazio e `'*'` que indica a presença de uma mina. As dimensões do tabuleiro podem mudar em cada caso de teste, por este motivo o seu programa deverá detectar o tamanho do tabuleiro levando em consideração que os caracteres estão posicionados em sequência, i.e., **não** existe nenhum tipo de separação entre eles como espaço ou vírgula. Além disto, ao fim de cada linha do tabuleiro existe um `'\n'`. Um exemplo de um arquivo contendo um tabuleiro é:

```
....*_
*....._
......_
.*...._
....._
```

Atente para o fato que o caractere `'_'` apenas simboliza a presença de uma quebra de linha (caractere `'\n'`) e **não** está de fato presente no arquivo de entrada.

3.2 Opções

Como explicado anteriormente, o sistema deverá possuir comportamentos distintos de acordo com a opção de entrada.

3.2.1 Opção: 1

O seu programa deverá imprimir no `stdout` o tabuleiro que foi carregado do arquivo de entrada. Os caracteres devem sem impressos em sequência, i.e., sem qualquer outro tipo de caractere de separação. Ao fim de cada linha uma quebra de linha deverá ser impressa, i.e., um `'\n'` deverá ser impresso. Assim, a saída produzida na execução do arquivo anterior seria:

```
....*_
*....._
......_
.*...._
....._
```

Novamente, atente para o fato que o caractere `'_'` apenas simboliza a presença de uma quebra de linha (caractere `'\n'`) e **não** deverá de fato estar presente na saída do seu programa.

3.2.2 Opção: 2

O seu programa deverá preencher o tabuleiro com as "dicas" numéricas. Essa dica é composta por um número que informa a quantidade de minas presentes na vizinhança 1-anel daquele ponto, i.e., os 8 quadrantes adjacentes (na vertical, horizontal e diagonal). Importante ressaltar que esta dica numérica só deverá estar presente nos quadrantes que inicialmente estavam vazios, i.e., tinham o caractere `'.'`. Um exemplo do preenchimento desta dica está na Figura abaixo:

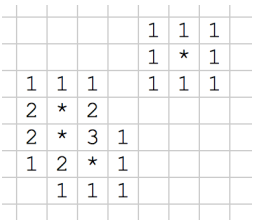

Figura 1: Tabuleiro preenchido com as dicas numéricas

Na Figura 1 é possível notar uma "borda" formada por dicas numérias que circunda todas as minas. Por exemplo a dica 3 revela ao jogador que na vizinhança 1-anel daquele quadrante existem 3 minas terrestres, como de fato é possível ve-las abaixo, do lado esquerdo e na diagonal superior esquerda.

Após a inicialização do tabuleiro com as dicas, o seu programa deverá imprimi-lo no `stdout`. Os caracteres devem sem impressos em sequência, i.e., sem qualquer outro tipo de caractere de separação. Ao fim de cada linha uma quebra de linha deverá ser impressa, i.e., um `'\n'` deverá ser impresso. Assim, a saída produzida na execução do arquivo de entrada anterior seria:

```
11.1*␣
*1.11␣
221..␣
1*1..␣
111..␣
```

De modo similar, o caractere `'␣'` apenas simboliza a presença de uma quebra de linha (caractere `'\n'`) e **não** deverá de fato estar presente na saída do seu programa.

3.2.3 Opção: 3

O seu programa deverá ler uma única posição x, y que informa a casa que deverá ser revelada. Assim, um exemplo de arquivo de entrada que executaria corretamente a opção 3 seria:

```
4
1.board
4 4
```

As coordenadas dos quadrantes serão fornecidas da seguinte maneira:

$$\begin{array}{cccc} [0,0] & [0,1] & \dots & [0,n] \\ [1,0] & [1,1] & \dots & [1,n] \\ \vdots & \vdots & \ddots & \vdots \\ [m,0] & [m,1] & \dots & [m,n] \end{array}$$

Dado um quadrante do tabuleiro este pode conter uma mina, uma dica ou um vazio. Caso o quadrante informado seja uma mina todo o tabuleiro deverá ser revelado, informando deste modo que o usuário perdeu o jogo. No exemplo do tabuleiro 1, caso a entrada informada seja $0, 4$ (corresponde à mina da primeira linha) o resultado que o programa deverá imprimir é:

```
11.1*␣
*1.11␣
221..␣
1*1..␣
111..␣
```

Outro possível cenário é o quadrante informado indicar uma dica numérica, neste caso somente o valor da dica deverá ser indicado. No exemplo do tabuleiro 1, caso a entrada informada seja $0, 0$ (corresponde à dica da primeira linha e primeira coluna) o resultado que o programa deverá imprimir é:

```
1XXXX␣
XXXXX␣
XXXXX␣
XXXXX␣
XXXXX␣
```

Note que os quadrantes que ainda não foram revelados devem ser impressos com um caractere 'X' (maiúsculo). Por fim, a situação mais complexa acontece quando o usuário seleciona um quadrante vazio. Neste caso todas as casas vazias adjacentes ao quadrante selecionado devem ser reveladas. Este processo de deve parar somente ao encontrar as bordas do tabuleiro ou algum número. Um exemplo deste processo pode ser visto na Figura 2. Supondo o cenário em que o quadrante cinza foi selecionado, o tabuleiro é revelado até que as bordas sejam encontradas ou um número seja revelado.

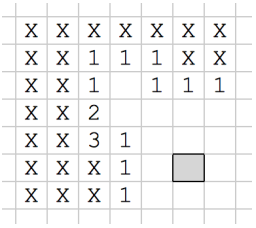

Figura 2: Algoritmo que revelar as casas vazias dado que o quadrante cinza foi selecionado.

No exemplo do tabuleiro 1, caso a entrada informada seja $4, 4$ (corresponde ao elemento vazio da ultima linha e ultima coluna) o resultado que o seu programa deverá imprimir é:

```
X1.1X_
X1.11_
X21.._
XX1.._
XX1.._
```

Em todos os três últimos cenários aqui descritos o caractere `'_'` apenas simboliza a presença de uma quebra de linha (caractere `'\n'`) e **não** deverá de fato estar presente na saída do seu programa.

4 IMPORTANTE

- utilize alocação dinâmica (memória heap) para armazenar os elementos da matriz
- não esqueça de liberar toda memória alocada antes de encerrar a execução do seu programa
- Crie quantas funções achar necessário, a modularização do sistema será levado em consideração no processo de correção.
- Não é permitido pesquisar na internet ou acessar qualquer site. Exceção para os 20 minutos finais que é permitido acessar o run.codes apenas para correção e validação do trabalho.
- Código da Disciplinas no run.codes: **2TSK**