Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: BCC201 – Introdução a Programação
Professores: Túlio A. M. Toffolo e Puca Huachi V. Penna

Trabalho Prático: *Campo Minado*

Quem nunca ouviu falar no jogo *Campo Minado*? Trata-se de um jogo popular criado em 1989 por Robert Donner, cujo objetivo é revelar as minas de um campo sem que estas explodam.

O jogo consiste de uma matriz em que cada célula contém um número ou uma mina. O número em uma célula indica a quantidade de minas adjacentes a esta célula. Em geral, o zero é omitido, ou seja, células vazias (sem valor) indicam que não há nenhuma mina adjacente. Note que cada célula tem no máximo 8 posições adjacentes.

Inicialmente os conteúdos das células ficam ocultas, e cabe ao jogador decidir qual célula revelar. Em cada jogada, o jogador tem duas opções:

1. Revelar uma nova célula.
2. Sinalizar a existência de mina em uma célula.

O jogo termina quando todas as células que não contém minas são reveladas. A Figura 1, extraída do site *Wikipedia*[1], apresenta um exemplo de jogo.

Figura 1: *Campo Minado*

O Trabalho Prático

Você deve implementar um programa que lê de um arquivo os dados: (i) as dimensões $n \times m$ da matriz; e (ii) os valores de cada célula. Cada célula pode ter os seguintes valores: de **'0'** a **'8'**, indicando o número de bombas em células adjacentes, ou **'x'**, indicando que a célula contém uma bomba.

Segue o arquivo que descreve o jogo apresentado na Figura 1. Note que a extensão **.jogo** é utilizada para um arquivo de entrada.

entrada.jogo

```
9 9

0 1 1 1 0 1 x 1 0
0 1 x 1 0 2 2 2 0
0 1 1 1 0 2 x 2 0
0 0 0 0 0 2 x 3 1
0 0 0 0 0 1 2 3 x
0 1 1 1 0 0 2 x 3
0 1 x 1 0 0 2 x 3
1 2 1 1 0 0 1 2 x
x 1 0 0 0 0 0 1 1
```

[1] https://pt.wikipedia.org/wiki/Campo_Minado

Seu programa deve ser capaz de salvar um jogo em andamento, para que o jogador seja capaz de continuar o jogo mais tarde. Para isso, o programa deve gerar um arquivo indicando as decisões que o usuário tomou. Este arquivo contém: (*i*) as dimensões $n \times m$ da matriz; e (*ii*) as opções do usuário para cada célula, que podem ser:

- **'-'**: indica que a célula ainda não foi revelada.
- **'o'**: indica que o usuário abriu a célula.
- **'x'**: indica que o usuário marcou a célula como bomba.

Seguem exemplos de dois arquivos (note que a extensão **.txt** é utilizada). O primeiro (inicio.txt) descreve um jogo no início, ou seja, antes que o jogador faça qualquer jogada. O segundo arquivo (quase.txt), descreve um jogo em andamento, tal qual apresentado pela Figura 1.

inicio.txt

```
9 9

- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
```

quase.txt

```
9 9

o o o o o o o x o -
o o x o o o o o o -
o o o o o o o x o -
o o o o o o o x o -
o o o o o o o o o -
o o o o o o o o x -
o o x o o o o o x -
o o o o o o o o o -
x o o o o o o o o -
```

Execução do programa

O arquivo contendo a descrição do jogo (tal como no exemplo anterior) deverá ser passado como um argumento na execução do programa.

Seu programa, logo após ser executado, deverá imprimir o *Campo Minado* (inicialmente cheio de **'-'**). Em seguida, o usuário será convidado a **digitar um comando** (ver possíveis comandos a seguir).

Comando	Argumento	Resultado
x	AC	Indica que a célula da linha A e coluna C contém uma mina.
o	EC	Revela o valor da célula da linha E e coluna C.
resolver		Resolve o jogo, marcando automaticamente as células com o valor certo.
salvar	out	Salva o jogo tal como está no momento no arquivo "out.txt". O programa deve salvar também o arquivo "out.jogo" com a descrição do jogo.
sair		Encerra o programa (sem salvar as últimas alterações).

Note que quando o usuário revelar uma célula cujo valor é zero, todas as células adjacentes também devem ser reveladas! Após a execução de um comando, exceto **salvar** e **sair**, seu programa deve imprimir o *Campo Minado* atualizado.

Importante: seu programa deve avisar ao usuário quando ele atingir uma bomba, e quando ele concluir o jogo. O usuário também deve ser alertado com uma mensagem de erro caso faça uma alocação que infrinja as regras do jogo. Por exemplo, se o usuário digitar um comando inválido ou

tentar preencher uma célula que não existe, você deve informá-lo qual engano ele cometeu e solicitar que ele digite um comando novamente.

Exemplos de execução do programa serão disponibilizados em breve.

Desafio (1)

- Quando o usuário revelar uma célula com valor zero, o ideal é que não somente as células adjacentes sejam reveladas, mas também as adjacentes das adjacentes até que as células com valores diferentes de zero fiquem visíveis. Tente digitar "campo minado" no *Google* para jogar o jogo, e veja como funciona.
- O primeiro desafio é implementar este recurso de forma adequada!

Desafio (2)

- Se o usuário não especificar um arquivo de texto com o jogo de entrada, crie um jogo aleatório.
- O segundo desafio é gerar jogos interessantes, de diferentes níveis de dificuldade, de forma aleatória e gerar um arquivo texto contendo o jogo gerado!

Instruções

- O problema deve ser resolvido por meio de um programa em C ou C++.
- Não serão aceitos trabalhos que caracterizem cópia (mesma estrutura e algumas pequenas modificações) de outro.
- Após a entrega dos trabalhos serão marcadas entrevistas com cada um dos alunos para apresentação dos mesmos para um dos professores da disciplina.

Entrega

- A entrega do código-fonte será feita pelo Moodle em duas partes (datas limites serão definidas em breve).
- Também deverá ser entregue um breve relatório (impresso e em mãos) sobre o trabalho contendo:
  - Descrição do problema tratado (não copiar este enunciado).
  - Relato das dificuldades encontradas durante a realização do trabalho e soluções encontradas.
  - Referências de sites e outros materiais utilizados para confecção de trabalhos, incluindo consultas a colegas (especificar quais).
- Atenção: o código-fonte do programa não deve ser incluído no relatório.

Avaliação

- Funcionamento adequado do programa.
- Atendimento ao enunciado do trabalho.
- Clareza do código (que deve ser devidamente comentado e indentado).
- Utilização de funções.
- Adequação da estrutura do programa (variáveis e comandos utilizados).

- Apresentação do trabalho e relatório.
- Compilação (códigos que não compilam serão zerados, e *warnings* diminuirão a nota).