Programação I

Trabalho Prático 2020/2021

(v1.1)

→ Minesweeper ←

1. Minesweeper

O Minesweeper[1] é um puzzle para um único jogador. O objetivo do jogo é "limpar" um campo de minas sem detonar nenhuma delas, com ajuda de pistas sobre o número de minas vizinhas.

As minas estão espalhadas por um tabuleiro retangular dividido em células. No início do jogo todas as células estão cobertas. O jogo é jogado revelando, iterativamente, o conteúdo de uma célula à escolha. Se a célula revelada contiver uma mina o jogador perde o jogo; caso contrário, é exibido um algarismo na célula que indica quantas células adjacentes contêm minas. Se não houver minas adjacentes, a célula fica em branco e todos os quadrados adjacentes serão revelados recursivamente. O jogador usa esta informação para deduzir o conteúdo de outras células, podendo revelar, com segurança, cada célula ou marcá-la como contendo uma mina.

O jogo termina quando for revelada uma mina (jogo perdido) ou quando todas as células livres de minas forem reveladas (jogo ganho).

2. Descrição do trabalho

Com este trabalho pretende-se uma implementação em **linguagem C** para o jogo Minesweeper.

Em cada instante do jogo cada célula do tabuleiro está num de três estados possíveis: **descoberto**, **coberto** ou **sinalizado**. Uma célula coberta é identificada um "`.`", enquanto numa célula descoberta é exposto o seu "conteúdo" (nº de minas em células adjacentes, ou vazio caso este número seja zero). As células sinalizadas como contendo uma mina são identificadas com um "`f`" (ver Exemplos de interação).

Caso seja solicitada uma ação não permitida, esta deve ser ignorada. Exemplos de ações não permitidas são:

- escolher uma célula "fora" do tabuleiro ou uma ação inexistente
- cobrir ou sinalizar uma célula descoberta
- sinalizar uma célula excedendo o máximo de sinais disponíveis

Pretende-se que o programa permita jogar quer com tabuleiros gerados de um modo aleatório (mas satisfazendo as restrições dadas pelo utilizador), quer com tabuleiros lidos a partir de ficheiros de texto.

[1] https://en.wikipedia.org/wiki/Minesweeper_(video_game).

Depois de gerado ou lido o tabuleiro, em cada jogada é escolhida uma célula e uma operação a realizar (uncover, flag). O tabuleiro é atualizado após a realização da operação; se o jogo não terminar é iniciada uma nova jogada.

O ficheiro de texto com o tabuleiro deve ter o seguinte formato:

- na primeira linha é indicada a configuração do tabuleiro: a dimensão (linhas e colunas por esta ordem) e o número de minas;
- nas linhas seguintes é indicada a posição de cada mina, uma por linha.

O conteúdo de um desses ficheiros poderia ser, por exemplo,

```
8 8 10
1 E
3 D
3 G
5 C
5 E
6 C
6 G
8 A
8 C
8 G
```

3. Exemplos de interação

Quando o jogo termina deve ser mostrada uma das seguintes mensagens:

- `Game Over! You LOST...`
- `You WIN!!!`

Para ilustrar a interação com programa, vejamos alguns exemplos em que **In** representa *input* do utilizador e **Out** o *output* do programa:

Exemplo 1: geração do tabuleiro

```
0 - Exit
1 - Generate board
2 - Read board
1
Insert, separated by space, the number of lines, columns and mines.
For instance: 10 15 5
8 8 8

   A B C D E F G H
 1 . . . . . . . .
 2 . . . . . . . .
 3 . . . . . . . .
 4 . . . . . . . .
 5 . . . . . . . .
 6 . . . . . . . .
 7 . . . . . . . .
```

```
 8 . . . . . .
```

```
Flags left: 8
State the position and the action: u(ncover) or f(lag).
For instance: 1 A u
1 A u
```

```
   A B C D E F G H
 1         1 . . .
 2   1 1 2 2 . . .
 3   1 . . . . . .
 4   1 1 . . . . .
 5     1 . . . . .
 6     1 . . . . .
 7 1 1 3 . . . . .
 8 . . . . . . . .
```

```
Flags left: 8
State the position and the action: u(ncover) or f(lag).
For instance: 1 A u
3 C f
```

```
   A B C D E F G H
 1         1 . . .
 2   1 1 2 2 . . .
 3   1 f . . . . .
 4   1 1 . . . . .
 5     1 . . . . .
 6     1 . . . . .
 7 1 1 3 . . . . .
 8 . . . . . . . .
```

```
Flags left: 7
State the position and the action: u(ncover) or f(lag).
For instance: 1 A u
3 C f
```

```
   A B C D E F G H
 1         1 . . .
 2   1 1 2 2 . . .
 3   1 . . . . . .
 4   1 1 . . . . .
 5     1 . . . . .
 6     1 . . . . .
 7 1 1 3 . . . . .
 8 . . . . . . . .
```

```
Flags left: 8
State the position and the action: u(ncover) or f(lag).
For instance: 1 A u
8 G u
```

```
   A B C D E F G H
```

```
 1          1 * . .
 2    1 1 2 2 . . .
 3    1 * . * . . .
 4    1 1 . . . . *
 5      1 . . . . .
 6      1 * . . . .
 7 1 1 3 . . . . .
 8 . * . * . . * .

Game Over! You LOST…

0 - Exit
1 - Generate board
2 - Read board
```

Exemplo 2: leitura do tabuleiro

```
0 - Exit
1 - Generate board
2 - Read board
2
Insert the file name
mines_8_8_10.txt

Initial board
   A B C D E F G H
 1 . . . . . . . .
 2 . . . . . . . .
 3 . . . . . . . .
 4 . . . . . . . .
 5 . . . . . . . .
 6 . . . . . . . .
 7 . . . . . . . .
 8 . . . . . . . .

Flags left: 10
State the position and the action: u(ncover) or f(lag).
For instance: 1 A u
1 A u

   A B C D E F G H
 1          1 . . .
 2    1 1 2 2 . . .
 3    1 . . . . . .
 4    1 1 . . . . .
 5      1 . . . . .
 6      1 . . . . .
 7 1 1 3 . . . . .
 8 . . . . . . . .

Flags left: 10
State the position and the action: u(ncover) or f(lag).
For instance: 1 A u
```

4. Implementação

Para o desenvolvimento do trabalho assuma que o tamanho máximo do tabuleiro é de 26x26 células (atenção que o número que identifica a linha deve ter 2 caracteres, sendo o primeiro um espaço no caso de ser menor que 10).

Comece por decidir qual a representação de uma célula e do tabuleiro. **A variável que representa o tabuleiro deve ser declarada na função `main()`.**

Depois estruture o seu programa em funções. Deverá, obrigatoriamente, implementar as seguintes:

- `generate_board( board, nlines, ncols, nmines )`
  - preenche um tabuleiro (parâmetro `board`) de dimensão `nlin` linhas e `ncol` colunas com `nmines` minas. Utilize a função **`rand()`** (`stdlib.h`) para gerar a posição de cada mina
- `print_board( board, nlines, ncols, showmines )`
  - mostra o tabuleiro no *standard output* (ecrã). O parâmetro `showmines` indica se as minas devem ser mostradas (com o carácter `*`).
- `uncover( board, nlines, ncols, line, col )`
  - descobre a célula (`line`, `col`) de acordo com as regras indicadas na descrição do jogo.
- `flag( board, nlines, ncols, line, col )`
  - sinaliza a célula (`line`, `col`).

Sugere-se a organização do código nos seguintes ficheiros:

- `mine_functions.h`: com os protótipos das funções do programa
- `mine_functions.c`: implementação das funções definidas em `mine_functions.h`
- `minesweeper.c`: implementação da função `main()`

O comando

```
gcc -c minesweeper.c
```

compila o ficheiro `minesweeper.c` e gera o ficheiro objeto `minesweeper.o`

O comando

```
gcc -o minesweeper minesweeper.o mine_functions.o
```

liga os ficheiros objeto (`minesweeper.o` e `mine_functions.o`) e cria o executável `minesweeper`

5. Condições Gerais

- O trabalho deverá ser efetuado por grupos de 2 elementos. Em casos excecionais, e acordados com os docentes, poderá ser feito por apenas 1 elemento.
- A entrega será via Moodle e até ao final do dia lá indicado.
- O trabalho será discutido em dia e horário a anunciar.

- O relatório deverá conter a identificação dos elementos do grupo assim como uma descrição sucinta quer dos tipos das principais variáveis (incluindo: tabuleiro, célula), quer das principais funções (incluindo: uncover, cover, flag). Preferencialmente não deve ter código.

- O trabalho será classificado entre 0 e 20 valores e serão factores de avaliação:

    - a clareza dos algoritmos implementados;

    - a correção do código;

    - a legibilidade do código (a qualidade dos identificadores, os comentários que o acompanham, etc);

    - a qualidade do relatório (lendo-o, deve ficar-se com uma ideia precisa do funcionamento do trabalho).

- Será aplicado o código de conduta do Departamento de Informática. Em caso de fraude, para além reprovação à disciplina, a situação será reportada.

6. Entrega

- O upload do trabalho tem de ser efectuado através do Moodle e **só** deverá ser efectuado pelo elemento do grupo que tiver o **número menor**.

- Deverá ser um ficheiro **`.tar.gz`** ou **`.zip`** que ao descomprimir crie uma pasta **`num1_num2`** em que `num1` e `num2` são os números dos alunos que compõem o grupo por ordem crescente, por exemplo: `123_456`. Nessa pasta deverá existir uma subpasta **`src`** com todo o código fonte necessário para testar o programa. Deverá também existir na directoria principal um ficheiro (**`relatorio`**) em formato texto, HTML ou PDF com o relatório.