RebelWars

Introdução à Programação para a Ciência e Engenharia

Enunciado do Projecto, versão 1.2 – 22-11-2022

Contacto: carla.ferreira@fct.unl.pt

Alterações em relação à versão 1.0:

- A posição do canto superior esquerdo passou a ter a coordenada (0,0) em vez de (1,1).
- Clarificada as situações de erro nos comandos path e prisoners.

Data de entrega: até às 23h59 do dia 19 de Dezembro de 2022.

Realização em grupos: este projecto deve ser realizado por grupos de 3 alunos.

Entregáveis: Submissão e aceitação do código fonte ao **Mooshak** (pontuação > 0), através de um ficheiro zip. Ter em atenção que no registo no Mooshak o nome do grupo tem que seguir o seguinte formato "`XXXXX_YYYYY_ZZZZZ`", com `XXXXX < YYYYY < ZZZZZ`, onde `XXXXX`, `YYYYY` e `ZZZZZ` representam os números dos membros do grupo. Ver a página da disciplina para obter mais detalhes sobre como submeter projectos ao Mooshak.

Recomendações: Valorizamos a documentação do código fonte, assim como a utilização do melhor estilo de programação possível e, claro, o correcto funcionamento do projecto. Comente cuidadosamente as suas classes. Os alunos podem e devem discutir quaisquer dúvidas com a equipa docente, bem como discutir com os seus colegas este projecto, mas **não podem partilhar o seu código com outros colegas**. Este projecto deve ser conduzido com total respeito pelo Código de Ética disponível na página da disciplina.

1. Desenvolvimento da aplicação RebelWars

1.1. Descrição do problema

O objectivo deste trabalho é desenvolver um jogo de computador para ajudar os rebeldes a coleccionar armas para o exército da Aliança Rebelde. Os heróis do jogo são os rebeldes que estão a ser perseguidos pelos *stormtroopers* (vilões). Os personagens deslocam-se num labirinto que é um espaço bidimensional. Os movimentos dos rebeldes são controlados pelo jogador, enquanto os *stormtroopers* obedecem cegamente a ordens superiores (são controlados pelo computador).

Os rebeldes deslocam-se no labirinto colectando armas que foram escondidas em vários locais do planeta Quesh e tentando não ser capturados pelos *stormtroopers.* **O jogo termina quando os rebeldes coleccionaram todas as armas ou foram todos capturados pelos *stormtroopers*.** Existem localizações que têm uma poção escondida (*adrenals*) que dá ao rebelde poderes especiais (mais à frente vamos ver quais).

Os personagens deslocam-se numa de 4 direcções: cima, baixo, esquerda e direita. Não existem deslocações nas diagonais. Nenhum dos personagens atravessa paredes.

A aplicação tem uma funcionalidade para a leitura do labirinto. Após a leitura do labirinto é possível adicionar vários rebeldes. Os heróis podem ser colocados numa qualquer posição vazia do labirinto. O número máximo de heróis que podem ser adicionados ao jogo está limitado pelo número de posições livres do labirinto. Pode considerar que o labirinto tem pelo menos uma posição livre, permitindo adicionar pelo menos um soldado rebelde.

O labirinto é representado por um rectângulo com `l` linhas e `c` colunas. Pode assumir que o labirinto será sempre um rectângulo. O canto superior esquerdo tem a coordenada **(0, 0)** e o canto inferior direito a **(`l-1, c-1`)**. Cada carácter destas linhas tem um significado:

- `'#'` – parede do labirinto;
- `' '` – posição vazia no labirinto;
- `'.'` – posição com uma arma;
- `'P'` – posição com uma poção;
- `'S'` – posição com um *stormtrooper*;
- `'R'` – posição com um rebelde;
- `'X'` – posição com um rebelde após tomar a poção.

Quando o labirinto é lido, este inclui todos os caracteres apresentados acima excepto os dois últimos que representam os soldados rebeldes. Depois de ler o labirinto o jogador pode criar os heróis do jogo. O jogo pode ser iniciado se o labirinto já tiver sido lido e existir pelo menos um soldado rebelde, ou seja, a fase de *setup* já tiver sido completada. Após iniciar o jogo já não é possível adicionar mais rebeldes.

Estando o jogo iniciado o jogador tem que dar instruções de movimentação aos rebeldes. Se um soldado rebelde se cruzar com um *stormtrooper* é capturado. Se passar por uma posição com uma arma, a arma é colectada e são-lhe atribuídos 10 pontos. Quando a arma é

colectada essa posição passa a estar vazia. Os *stormtroopers* não têm qualquer efeito sobre as posições com armas (porque as armas estão camufladas).

As personagens capturadas não voltam a ser mostradas no labirinto, mas deve ser possível consultar a sua informação.

A poção dá poderes especiais ao rebelde que a encontrar, mas não tem qualquer efeito nos *stormtroopers*. Após beber a poção esse soldado passa a ser representado pelo símbolo `'X'`, e pode capturar *stormtroopers* até o final do jogo. Após a poção ser consumida, essa posição passa a estar vazia.

O labirinto seguinte tem 9 linhas e 27 colunas, 2 poções mágicas, 10 *stormtroopers* e 10 localizações vazias onde se podem colocar rebeldes:

```
.....SSS.......SSS.........
.......  ......   ........#
P######  ######   #######P#
..#......#...........#....#
..###....#...........#....#
..#......#...........#....#
..#.......######.....#....#
.............................
...........SSSS.............
```

Para o labirinto acima os identificadores dos *stormtroopers* seguem uma ordenação primeiro por linha e depois por coluna. Na primeira linha, começando pela esquerda, estão os *stormtroopers* "ST-1", "ST-2", "ST-3", "ST-4", "ST-5" e "ST-6". Na última linha estão os *stormtroopers* "ST-7", "ST-8", "ST-9" e "ST-10". Cada *stormtrooper* tem um identificador único, o estado (activo ou capturado) e os rebeldes capturados até altura. Cada rebelde tem um nome único, o estado (activo, poção activa, capturado), a posição corrente no labirinto, os pontos obtidos e o trajecto que percorreu no labirinto.

Não há limite ao número de rebeldes (*stormtroopers*) que um stormtrooper (*rebelde*) pode capturar.

Como já foi dito os *stormtroopers* são controlados pelo computador. Na prática cada *stormtrooper* faz repetidamente uma sequência de movimentos pré-definidos. **Em cada momento, o *stormtrooper* vai tentar seguir o movimento que estiver selecionado na sua sequência, apenas se não conseguir mover-se passa ao movimento seguinte na sequência.** Note que cada *stormtrooper* individual, consoante a sua localização no labirinto, pode estar em estados diferentes.

Stormtrooper	**Sequência de movimentos**			
`S`	cima	baixo	esquerda	direita

A aplicação deve então permitir:

1. **Mostrar os comandos do jogo** (comando **`help`**). Este comando não requer quaisquer argumentos e é sempre bem-sucedido. O cenário seguinte ilustra a sua utilização.

```
help
upload: leitura do labirinto do jogo
print: mostrar labirinto do jogo
rebel: adicionar soldado rebelde
start: iniciar o jogo
move: mover as personagens
rebels: listar soldados rebeldes
stormtroopers: listar stormtroopers
path: listar caminho percorrido por soldado rebelde
prisoners: listar rebeldes capturados por stormtrooper
reset: reiniciar o jogo
help: mostrar os comandos do jogo
exit: sair do jogo
```

2. **Leitura do labirinto do jogo** (comando **`upload`**). São fornecidos dois inteiros **`l`** e **`c`**, com a altura e largura do labirinto. Seguem-se **`l`** linhas, cada uma delas com **`c`** caracteres, representando o conteúdo do labirinto. Em caso de sucesso é apresentada a mensagem (`Maze accepted.`).
   A operação falha se: (1) já existir um labirinto definido (`Maze already defined.`).
3. **Mostrar labirinto do jogo** (comando **`print`**). Imprime o labirinto, no seu estado actual, incluindo informação sobre a localização dos personagens caso existam. Por exemplo, o labirinto deve apresentar carácter `'S'` na posição (1,2) se existir nessa posição um *stormtroopers.* Em caso de sucesso, o labirinto é apresentado. Caso o jogo já tenha sido iniciado, o labirinto é seguido de uma linha com a mensagem (`Points: <N> Timer: <N> Rebels: <N> Game: <STATE>.`), que mostra o número de pontos actuais, o tempo corrente, o número de rebeldes ainda activos e o estado do jogo (`SETUP`, `ON` ou `OVER`).
   A operação falha se: (1) não existir um labirinto definido (`Maze is undefined.`).
4. **Adicionar soldado rebelde** (comando **`rebel`**). São fornecidos o nome do rebelde e dois inteiros, a linha e a coluna, indicando a posição onde o personagem será criado. Em caso de sucesso esta operação adiciona o rebelde ao jogo (`Rebel added.`). O número de rebeldes está limitado pelo número de posições vazias do labirinto.
   A operação falha se: (1) não existir um labirinto definido (`Maze is undefined.`); (2) o jogo já tiver sido iniciado (`Game setup has already finished.`); (3) já existir um rebelde com o nome dado (`Rebel name already exists.`); (4) a

posição do labirinto estiver fora do labirinto ou não for uma posição vazia (`Invalid maze position.`).

5. **Iniciar jogo** (comando **`start`**). Este comando não recebe qualquer parâmetro. Em caso de sucesso, o jogo é iniciado e mostra a mensagem (`Points: <N> Timer: <N> Rebels: <N> Game: <STATE>.`), que mostra o número de pontos actuais (`0`), o tempo corrente (`0`), o número de rebeldes activos, e o estado do jogo (`ON`).
   A operação falha se: (1) não existir um labirinto definido (`Maze is undefined.`); (2) o jogo já tiver sido iniciado (`Game setup has already finished.`); (3) não existir nenhum soldado rebelde no jogo (`Game needs a rebel.`).
6. **Mover personagens** (comando **`move`**). É fornecido o tipo de movimento (`U`, `D`, `L`, `R`) para cada um dos rebeldes ainda activos, segundo a ordem com que foram criados. Ou seja, têm de ser dados tantos movimentos quantos os rebeldes activos. Em caso de sucesso, esta operação movimenta todos os personagens, primeiro os rebeldes e depois os *stormtroopers*, ambos pela ordem que foram criados. Também incrementa o tempo por uma unidade e gera o output (`Points: <N> Timer: <N> Rebels: <N> Game: <STATE>.`), onde `<STATE>` representa o estado do jogo (`ON` ou `OVER`).
   Pode não ser possível mover um rebelde, porque a posição destino é uma parede ou está ocupada por outro soldado rebelde. O mesmo pode acontecer a um *stormtrooper*, se não se puder movimentar das quatro direcções. Se o rebelde não se movimentar o número de passos não é alterado.
   A operação falha se: (1) não existir um jogo activo (`Game is not on.`).
   São sempre fornecidos um número de movimentos que corresponde ao número de rebeldes activos.
7. **Listar personagem** (comando **`rebels`** ou **`stormtroopers`**). Em caso de sucesso, a operação produz o cabeçalho (`Rebels:`), ou (`Stormtroopers:`), respectivamente. O cabeçalho é seguido pela listagem dos vários personagens apresentando em cada linha o nome da personagem, a posição no labirinto, o seu estado (`SUPERCHARGED`, `ACTIVE` ou `CAPTURED`) e no caso dos rebeldes apresenta o número de pontos colectados. Nas listagens os personagens devem ser apresentados por ordem de inserção.
   A operação falha se: (1) não existir um labirinto definido (`Maze is undefined.`); (2) não existirem personagens a listar (`Nothing to list.`).
8. **Listar caminho percorrido por soldado rebelde** (comando **`path`**)**.** É fornecido o nome do soldado rebelde. Como resultado a operação apresenta o caminho percorrido, da posição mais antiga à mais recente. A listagem deve ser precedida por um cabeçalho (`Rebel <NAME> has taken <N> steps and is <STATE>:`), onde `<NAME>` é o nome do rebelde, `<N>` é o número de movimentos totais e `<STATE>` é `ACTIVE`, `SUPERCHARGED` ou `CAPTURED`. De seguida as posições percorridas são apresentadas, uma por linha. Cada linha tem o formato (`<row> <col>`).

A operação falha se: (1) o jogo não estiver no estado `ON` ou `OVER` (`Game has not started.`); (2) não existir um rebelde com o nome dado (`Rebel does not exist.`).

9. **Listar rebeldes capturados por *stormtrooper*** (comando **`prisoners`**). É fornecido o identificador do *stormtrooper*. Como resultado a operação apresenta, pela ordem de captura, o nome do rebelde e a localização onde foi capturado. A listagem deve ser precedida por um cabeçalho (`Stormtrooper <ID> has captured <N> rebels and is <STATE>:`) , onde `<ID>` é o identificador do *stormtrooper*, `<N>` é o número rebeldes capturados e `<STATE>` é `ACTIVE` ou `CAPTURED`. De seguida os rebeldes capturados são apresentados, um por linha. Cada linha deve apresentar o nome do rebelde e a posição onde foi capturado.
   A operação falha se: (1) o jogo não estiver no estado `ON` ou `OVER` (`Game has not started.`); (2) não existir um *stormtrooper* com o identificador dado (`Stormtrooper does not exist.`).
10. **Reiniciar o jogo** (comando **`reset`**). Reinicia o jogo. Este comando tem sempre sucesso e pode ser executado em qualquer estado do jogo. Após a execução deste comando, terá que ser feito novamente o setup do jogo (leitura do labirinto e adição de rebeldes). Em caso de sucesso, o jogo é iniciado e mostra a mensagem (`Game was reset.`).
11. **Sair do jogo** (comando **`exit`**). A operação tem sempre sucesso (`Exiting.`).

1.2. Exemplo de interacção com a aplicação

A aplicação desenvolvida tem que garantir o modelo de interacção ilustrado no exemplo seguinte:

```
print
Maze is undefined.
upload
9 27
.....SSS.......SSS.........
.......  ......   .........
P######  ######   #######P.
..#......#..........#.....
..###....#..........#.....
..#......#..........#.....
..#.......######.....#.....
...........................
...........SSSS............
Maze accepted.
upload
2 4
#...
O...
Maze already defined.
rebel
Jyn
2 7
Rebel added.
rebel
Cassian
2 15
Rebel added.
start
Points: 0 Timer: 0 Rebels: 2 Game: ON.
move
D D
Points: 20 Timer: 1 Rebels: 2 Game: ON.
print
.....   .......   .........
.....SSS.......SSS.........
P######  ######   #######P.
..#....R.#.....R.....#.....
..###....#..........#.....
..#......#..........#.....
..#.......######.....#.....
...........SSSS............
...........    ............
Points: 20 Timer: 1 Rebels: 2 Game: ON.
rebels
Rebels:
Jyn 3 7 ACTIVE 10
Cassian 3 15 ACTIVE 10
path
```

```
Jyn
Rebel Jyn has taken 1 steps and is ACTIVE:
2 7
3 7
exit
Exiting.
```

2. Desenvolvimento

Pode começar por desenvolver a função `main` que inclui o interpretador de comandos do seu programa, identificando claramente quais os comandos que a sua aplicação deve suportar, e as suas entradas e saídas.

Depois, terá de identificar as entidades necessárias para a implementação deste sistema. **Especifique cuidadosamente as classes que necessitará**. Deverá documentar adequadamente o seu código.

Deve começar por construir um esqueleto da sua função principal. Numa fase inicial, o seu programa não irá fazer muito. Lembre-se da regra da versão estável: não tente fazer tudo ao mesmo tempo. Construa o seu programa de forma incremental, e teste os incrementos à medida que constrói as novas funcionalidades do sistema.

Analise com cuidado os ficheiros de teste, quando estes estiverem disponíveis. Deve começar com um sistema básico implementando os comandos **help** e **exit**, que são suficientes para verificar se o seu interpretador de comandos está a funcionar correctamente. Depois, deve implementar o comando para ler (**upload**) e mostrar (**print**) o labirinto. Passo a passo, deverá incrementar as funcionalidades do seu programa e testá-las. Não tente fazer todas as funcionalidades ao mesmo tempo. É uma péssima ideia!

3. Submissão ao Mooshak

Para submeter o projecto ao Mooshak, deve registar o seu grupo no concurso Mooshak IPCE-22-23-Proj e seguir as instruções que estão disponibilizadas na página da disciplina no Moodle (o concurso Mooshak estará disponível a 25 de Novembro). Tal como nos TPCs será avaliada a última submissão do projecto.

3.1. Sintaxe dos comandos

Para cada comando, o programa produzirá apenas um output. As condições de erro de cada comando têm de ser verificadas exactamente na mesma ordem que está descrita neste documento. Se uma dessas condições for detectada, não é necessário verificar as restantes condições, uma vez que apenas deve ser apresentada a mensagem correspondente à primeira condição de erro. Por exemplo, se ao executar o comando **rebel** já existir um soldado rebelde com o nome dado, a mensagem de saída deve ser "`Rebel name already exists.`" e não é necessário verificar se a posição inicial para a personagem é válida.

3.2. Testes

Os testes Mooshak verificam de forma incremental a implementação dos comandos. Os testes vão ser publicados a 25 de Novembro.

Quando os ficheiros de teste estiverem disponíveis, utilize-os para testar os comando que já implementou, e comece a submeter a implementação parcial do projecto ao Mooshak. Faça-o desde o início, mesmo que apenas tenha implementado os comandos de saída e de ajuda. Boa sorte!