Ciências ULisboa | Informática

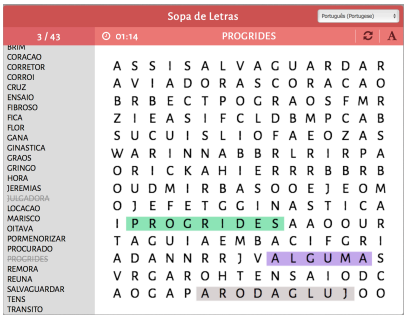
Introdução à Programação

Licenciatura em Engenharia Informática

Trabalho: 1ª parte

2020/2021

Sopa-de-Letras



O trabalho de programação que vos é proposto em IP é em torno do jogo *Sopa-de-Letras*. Cada partida de *Sopa-de-Letras* é jogada numa quadrícula como a da figura, onde estão "escondidas" várias palavras. Em cada jogada, o jogador seleciona um conjunto de posições contíguas da quadrícula. Se as letras nessas posições formam uma das palavras escondidas, essa palavra é considerada encontrada. No exemplo ao lado, podemos ver que já foram encontradas três palavras. O jogo pode ser jogado com tempo limitado ou sem tempo, caso em que apenas termina quando todas as palavras escondidas foram encontradas.

A primeira fase do trabalho vai servir para se familiarizarem com o jogo, obrigando-vos a pôr em prática os conhecimentos entretanto adquiridos.

Em que consiste o trabalho, afinal?

Nesta fase vamos considerar uma variante do jogo em que a quadrícula está preenchida com dígitos (em vez de letras) e em que temos de procurar certas sequências de dígitos (em vez de palavras) nas linhas da quadrícula. Os dígitos que podem ser usados são sempre os mesmos —os dígitos de 1 a 9— mas o tamanho da quadrícula é variável. As sequências tanto podem estar escondidas numa linha na ordem certa como na ordem inversa.

Usando um número inteiro para representar cada linha da quadrícula e cada sequência de dígitos escondida, a vossa tarefa é desenvolver código Java que ofereça funcionalidades úteis no contexto do jogo: saber se uma sequência está escondida numa linha e saber a sequência definida por uma jogada (intervalo de posições contíguas da linha).

Mais concretamente, devem implementar uma classe **`NumberSearch`** que inclua:

- A definição de uma função com o nome `digits` que, dado um inteiro `num`, que se assume ser positivo, devolve o número de dígitos de `num` (ex., `digits(456)=3`).
- A definição de uma função com o nome `reverseDigits` que, dado um inteiro `num`, que se assume ser positivo e com 9 dígitos no máximo, devolve o número com os dígitos na ordem inversa (ex., `reverseDigits(123)=321; reverseDigits(120)=21`).
- A definição de uma função com o nome `isSubsequence` que, dados dois números inteiros `num1` e `num2`, que se assumem ser ambos positivos, indica se a sequência de dígitos de `num1` é uma subsequência da sequência de dígitos de `num2` (ex., `isSubsequence(23,1234)=true`).
- A definição de uma função com o nome `subsequence` que, dados três números inteiros `num`, `from` e `to`, e assumindo que `num>0` e `1≤from≤to≤digits(num)`, devolve o número formado pelos

dígitos nas posições `from` até `to` de `num` (ex., `subsequence(1234,2,3)=23`).

- A definição de uma função com o nome `isValidRow` que, dado um número inteiro `num`, e um número inteiro `numberDigits` que se assume ser positivo, verifica se: `num` é positivo, tem `numberDigits` dígitos e é composto por dígitos entre `1` e `9`.
- A definição de uma função com o nome `isValidSequence` que, dado um número inteiro `num`, e um número inteiro `numberDigits` que se assume ser positivo, verifica se: `num` é positivo, tem no máximo `numberDigits` dígitos e é composto por dígitos entre `1` e `9`.
- A definição de um procedimento que, dado um número inteiro `numberDigits`, que se assume ser positivo e, dados dois números inteiros `row` e `sequence` procede da seguinte forma:
  1. Verifica se `row` é válido. No caso de `row` não ser válido, imprime uma mensagem com essa informação.
  2. Verifica se `sequence` é válido. No caso de `sequence` não ser válido, imprime uma mensagem com essa informação.
  3. Se `row` e `sequence` são ambos válidos, então imprime uma mensagem indicando se a `sequence` está ou não escondida em `row` (na ordem certa ou ordem inversa).
- A definição de um procedimento que, dado um número inteiro `row`, que se assume ser positivo e, dados dois números inteiros `from` e `to`, procede da seguinte forma:

  Verifica se o par `from,to` define um intervalo de posições válidas da sequência de dígitos de `row` (i.e., `1≤from≤to≤digits(row)`). Se forem válidos imprime uma mensagem indicando o número composto pelos dígitos nesse intervalo de posições. Se não forem válidos, imprime uma mensagem com essa informação.
- A implementação do método `main` que deverá começar com a declaração e inicialização do tamanho das linhas da quadrícula com o valor 9 e de seguida, teste os dois procedimentos com os seguintes dados:

`row` = 19874**5334**	`sequence` = 533	(dados válidos, ocorre)
`row` = 19874**5334**	`sequence` = 335	(dados válidos, ocorre)
`row` = 19874**5334**	`sequence` = 5334	(dados válidos, ocorre)
`row` = 198745334	`sequence` = 121	(dados válidos, não ocorre)
`row` = 19874533	`sequence` = 335	(`row` inválido)
`row` = 198745334	`sequence` = 305	(`sequence` inválido)
`row` = 19874533	`sequence` = 305	(ambos os dados inválidos)

`row` = 19874**5334**	`from` = 6 `to` = 8	(533)
`row` = **19874**5334	`from` = 1 `to` = 5	(19874)
`row` = 198745334	`from` = 8 `to` = 6	(`from-to` inválidos)
`row` = 198745334	`from` = 6 `to` = 12	(`from-to` inválidos)

O resultado da execução do programa deve ser semelhante ao mostrado abaixo:

```
$ java NumberSearch
The sequence 533 is hidden in row 198745334.
The sequence 335 is hidden in row 198745334.
The sequence 5334 is hidden in row 198745334.
The sequence 121 is not hidden in row 198745334.
The row 19874533 is not valid.
The sequence 305 is not valid.
The row 19874533 is not valid. The sequence 305 is not valid.

The sequence from position 6 to 8 in row 198745334 is 533.
The sequence from position 1 to 5 in row 198745334 is 19874.
The range from 8 to 6 is not valid in row 198745334.
The range from 6 to 12 is not valid in row 198745334.
```

Notem que 1) podem incluir na vossa classe outros procedimentos/funções que considerem úteis e 2) devem testar o vosso código com mais exemplos.

O que posso usar neste trabalho?

O objetivo deste trabalho é pôr em prática os conhecimentos lecionados até à aula teórica 8: utilização de tipos primitivos, estruturas básicas de controlo de fluxo e a abstração procedimental. Assim sendo, o código que vão escrever não pode usar tipos sofisticados (ainda não lecionados) como *String* e vetores.

O que entrego?

O ficheiro `NumberSearch.java` com a solução. Não há relatório a entregar porque o vosso software é a vossa documentação. Assim, não se esqueçam de comentar condignamente a vossa classe. Devem incluir no início da classe um cabeçalho *javadoc* com `@author` (nome e número dos alunos que compõem o grupo). Para cada procedimento/função definidos há que preparar um cabeçalho incluindo a sua descrição, e, se for caso disso, `@param`, `@requires`, `@ensures` e `@return`. Apresentem um texto "limpinho", que siga as normas de codificação em Java, bem alinhado e com um número de colunas adequado. Consultem, na página da disciplina, o *Guia de Estilo Java para IP*.

Como entrego o trabalho?

Um dos alunos do grupo entrega o trabalho através da ligação que, para o efeito, existe na página da disciplina no *moodle*. O prazo de entrega é dia 15 de Novembro às 23h55.

Quanto vale o trabalho?

Esta 1ª parte do trabalho é cotada para 5 valores e irá somar às notas das duas partes que se seguirão.