Teste prático TI Back End – Lógica de Programação

Pré-requisitos

Você pode realizar a prova em **UMA** das duas linguagens abaixo (escolha sua linguagem de preferência para resolver este desafio):

1. **C#**

O projeto roda na ferramenta de desenvolvedor chamada Visual Studio. Caso ainda não tenha, o instalador pode ser baixado através do seguinte link (versão *Community*): https://visualstudio.microsoft.com/pt-br/vs/

Uma vez que o Visual Studio está devidamente instalado, deve ser feito o download do projeto *template* através do seguinte link: https://www.atoscapital.com.br/template-solution-csharp.rar

2. **Java**

O projeto roda na ferramenta de desenvolvedor chamada Eclipse. Caso ainda não tenha, o instalador pode ser baixado através do seguinte link (Eclipse IDE): https://www.eclipse.org/downloads/

Uma vez que o *Eclipse IDE for Java Developers* está devidamente instalado, deve ser feito o download do projeto *template* através do seguinte link: https://www.atoscapital.com.br/template-solution-java.rar

Vídeo explicativo

É importante assistir o vídeo em que é explicado como rodar o projeto e o que se espera que você entregue. A explicação é feita usando o projeto *template* para a linguagem C#, contudo serve para qualquer uma das linguagens aceitas. Link do vídeo: https://youtu.be/Xb_MJ_q0oHU

Desafio do labirinto

Será dada uma entrada em arquivo texto, onde na primeira linha contém as dimensões do labirinto (Linhas Colunas) e nas demais linhas o labirinto em si, em que:

- 1 indica uma parede (isto é, não pode seguir neste ponto da matriz)
- 0 indica um caminho possível de se trafegar
- X é o ponto de partida (não necessariamente é um canto do mapa)

O objetivo é encontrar a única saída, sem "andar pelas paredes" e seguindo a seguinte ordem de prioridade (quando puder se deslocar):

1) Ir para cima (C)
2) Ir para a esquerda (E)
3) Ir para a direita (D)
4) Ir para baixo (B)

Caso se alcance um ponto em que não é possível se movimentar e/ou não tenham mais posições que ainda não foram visitadas, deve-se retornar usando o mesmo caminho utilizado até este ponto "sem-saída" até o último ponto onde teve mais de uma posição possível de movimento. A ordem de movimento só é utilizada quando há mais de uma posição possível de movimento para posições ainda não visitadas.

O desafio é elaborar um código-fonte dentro do projeto *Template* dentro da função **CodigoAtividade()** (se a prova for feita em C#) ou dentro da própria função **main** (se a prova for feita em Java) que seja capaz de:

1) ler o arquivo texto de entrada
2) identificar a dimensão da matriz do labirinto, em que o primeiro número indica o número de linhas e o segundo número indica o número de colunas (é separado por espaço)
3) identificar a posição de origem (ponto O localizado dentro da matriz). A posição "aumenta de valor" lendo de cima para baixo e/ou da esquerda para a direita. A posição na extremidade superior esquerda é a [1, 1] (linha 1 coluna 1) e a posição na extremidade inferior direita é a que representa o número de linhas e o número de colunas [L, C] (exemplo, se tem 4 linhas e 5 colunas, esta extremidade em questão é a [4, 5])
4) a partir do ponto de origem se deslocar (seguindo a ordem de prioridade de deslocamento) e encontrar a única saída (que se encontra no ponto 0 localizado em uma extremidade da matriz)
5) ao encontrar a saída gerar um arquivo texto de saída (na mesma pasta onde está o arquivo de entrada, só que com outro nome de arquivo. ex: *entrada.txt* é arquivo de entrada então o arquivo de saída pode ser *saída-entrada.txt*) contendo cada passo do trajeto, onde cada linha indica a direção e posição destinada. A primeira linha do arquivo de saída deve estar com O (origem) seguido da posição inicial

Para qualquer uma das linguagens aceitas, o projeto *template* já contém o código exemplo para ler o arquivo de entrada, identificar as dimensões da matriz e a posição de origem, assim como o código para escrever o trajeto no arquivo de saída. **O objetivo principal desta prova não é o conhecimento na linguagem em si, mas a habilidade de lógica de programação**.

Apesar do projeto *template* já conter uma forma de armazenar o labirinto e a posição original, fique à vontade para alterar e possivelmente criar novas classes, fazer uso de estruturas de dados conhecidas ou criar suas próprias, desde que faça tudo dentro da classe **frmAtividade.cs** (se a prova foi feita em C#) ou **Main.java** (se a prova foi feita em Java). **Não adicionar quaisquer outras bibliotecas ao projeto além das que já estão importadas** tendo em vista que o que você nos enviará por e-mail será o arquivo principal (**frmAtividade.cs** ou **Main.java**). Seguem os critérios decisivos para a escolha dos candidatos:

1) Conteúdo correto do arquivo de saída para as entradas que serão lidas
2) Boas práticas de programação
3) Tempo

Se houver algum tipo de empate que resulte em mais candidatos aprovados do que a quantidade de vagas ofertadas mesmo considerando os 3 critérios, a escolha será feita a critério da liderança da Atos Capital.

O desafio deve ser entregue até o dia definido pelo recrutador (no máximo às 23:59:59hrs do dia em questão), respondendo o e-mail que você recebeu com o conteúdo do desafio (**não use a opção RESPONDER A TODOS, use a opção RESPONDER para enviar apenas para o remetente, ignorando possíveis outros destinatários**), enviando em anexo apenas a classe **frmAtividade.cs** ou **Main.java**, que deve conter todo o código incluído para a solução do problema.

A seguir são apresentados três exemplos de entradas e suas respectivas saídas, independente da linguagem utilizada (que também estão no e-mail). Importante ressaltar que outras entradas também serão testadas. Qualquer dúvida, pode enviar e-mail que responderemos (adriano.maciel@atoscapital.com.br). Boa prova!

Exemplo 1:

Conteúdo do arquivo de entrada

```
5 8
1 1 1 1 1 1 1 1
1 1 0 1 0 1 1 1
1 1 0 0 0 1 1 1
X 0 0 1 0 0 0 0
1 1 1 1 1 1 1 1
```

Conteúdo correto do arquivo saída

```
O [4, 1]
D [4, 2]
D [4, 3]
C [3, 3]
C [2, 3]
B [3, 3]
D [3, 4]
D [3, 5]
C [2, 5]
B [3, 5]
B [4, 5]
D [4, 6]
D [4, 7]
D [4, 8]
```

Exemplo 2:

Conteúdo do arquivo de entrada

```
8 8
1 1 1 1 1 1 1 1
1 0 0 1 0 0 0 1
1 1 0 0 0 1 1 1
1 0 1 X 0 1 0 1
1 0 0 0 0 0 0 1
1 1 1 1 1 1 0 1
1 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1
```

ATOSCAPITAL
SERVIÇOS FINANCEIROS

Conteúdo correto do arquivo saída

O [4, 4]
C [3, 4]
E [3, 3]
C [2, 3]
E [2, 2]
D [2, 3]
B [3, 3]
D [3, 4]
D [3, 5]
C [2, 5]
D [2, 6]
D [2, 7]
E [2, 6]
E [2, 5]
B [3, 5]
B [4, 5]
B [5, 5]
E [5, 4]
E [5, 3]
E [5, 2]
C [4, 2]
B [5, 2]
D [5, 3]
D [5, 4]
D [5, 5]
D [5, 6]
D [5, 7]
C [4, 7]
B [5, 7]
B [6, 7]
B [7, 7]
E [7, 6]
E [7, 5]
E [7, 4]
E [7, 3]
E [7, 2]
D [7, 3]
D [7, 4]
D [7, 5]
D [7, 6]
D [7, 7]
D [7, 8]

Exemplo 3:

Conteúdo do arquivo de entrada

```
5 8
1 1 1 1 1 1 1 1
1 1 0 0 0 1 1 1
1 1 0 0 0 1 1 1
X 0 0 0 0 0 0 1
1 1 0 1 1 1 1 1
```

Conteúdo correto do arquivo saída

```
O [4, 1]
D [4, 2]
D [4, 3]
C [3, 3]
C [2, 3]
D [2, 4]
D [2, 5]
B [3, 5]
E [3, 4]
B [4, 4]
D [4, 5]
D [4, 6]
D [4, 7]
E [4, 6]
E [4, 5]
E [4, 4]
C [3, 4]
D [3, 5]
C [2, 5]
E [2, 4]
E [2, 3]
B [3, 3]
B [4, 3]
B [5, 3]
```