Labirintos

Seguindo a definição apresentada na wikipedia[1], um labirinto é um conjunto de percursos intrincados, criado com a intenção de desorientar alguém que o percorre. Neste VPL você deve desenvolver um programa que seja capaz de ler labirintos armazenados em arquivos (cada arquivo armazena um único labirinto) e determinar qual o caminho que leva até sua saída. Para efeitos de implementação, algumas simplificações são necessárias. Cada labirinto é especificado por um retângulo bidimensional com *n* linhas e *m* colunas, sendo cada um de seus elementos (ou células) uma parte que pode ser percorrida por um caminhante (representadas no arquivo de entrada por espaços em branco) ou uma parede (representa por caracteres diferentes de espaço). Assumindo que o canto superior esquerdo do labirinto é a célula `(0,0)`, você pode assumir que a entrada do labirinto sempre estará posicionada na célula `(1,0)` ao passo que a saída sempre estará posicionada na célula `(`*n*`-2, `*m*`-1)`. Um exemplo de labirinto é fornecido abaixo:

```
+---+---+-----+---+-------+-----+
    |   |     |   |       |     |
| --+ | | | | | | +-+ --+ +-- |
|     |   | |   | | | |     |   |
| +---+---+ +-+-+ | | +-----+ | |
| |       |   |   |   |     | | |
| +---- +-+-- | +-+ +-+ +-- | | |
|       |     | | | |   |   | | |
+---- +-+ +---+ | | | +-+ +-+ | |
|     |   |     |   | | |   |   | |
| +---+ +-+ +---+-+ | +-+ +-+ +-+
| |   | | | |     | |   |   |   |
| +-- | | | | | | +-+ | +-+ +-+ |
| |   | |   | | |   | |   |   | |
| | --+ | --+-+ +-- +-+-- +-- | |
| |     |       |         |     |
+-+-----+-------+---------+-----+
```

Seu programa deve receber como parâmetro via linha de comando o nome de um arquivo que contém um único labirinto. O arquivo deve ser então aberto e o labirinto lido. Seu programa deve então resolver o labirinto e mostrar a solução em tela, indicando o caminho que deve ser percorrido da entrada até a saída com o caractere “.” tal qual mostrado abaixo para o exemplo fornecido.

```
+---+---+-----+---+-------+-----+
..  |...|.....|...|       |     |
|.--+.|.|.| |.|.|.| +-+ --+ +-- |
|.....|...| |...|.| | |     |   |
| +---+---+ +-+-+.| | +-----+ | |
| |       |   |...|   |.....| | |
| +---- +-+-- |.+-+ +-+.+--.| | |
|       |     |.| | |...|...| | |
+---- +-+ +---+.| | |.+-+.+-+ | |
|     |   |.....|   |.|  .|   | |
| +---+ +-+.+---+-+ |.+-+.+-+ +-+
| |   | | |.|  ...| |...|...|   |
| +-- | | |.| |.|.+-+ |.+-+.+-+ |
| |   | |...| |.|...| |...|...| |
| | --+ |.--+-+.+--.+-+--.+--.| |
| |     |.......|  .......|  ....
+-+-----+-------+---------+-----+
```

Para a resolução deste problema crie tantas funções quanto desejar. É importante notar que este VPL foi feito especificamente para a prática de conceitos de recursão. Portanto, se sua solução não for recursiva, sua nota será zero, mesmo que seu programa resolva o problema corretamente. O arquivo de entrada contém somente um labirinto e não há quebra de linha (nova linha) após a última linha do labirinto. A impressão em tela do resultado deve seguir o mesmo padrão, ou seja, imprima o labirinto mas não imprima uma quebra de linha após sua impressão (não adicione um `\n` ao fim da última célula do labirinto).

Como mencionado anteriormente, seu programa deve receber o nome do arquivo de entrada como parâmetro via linha de comando. Para isso, sua função *main* deverá ter dois argumentos: *argc* e *argv*. A declaração da função *main* fica desta forma:

```
int main(int argc, char *argv[])
```

O argumento *argc* indica quantos parâmetros foram passados ao programa via linha de comando (do inglês: **arg**ument **c**ount), incluindo o próprio nome do programa. Quando você executa um programa na forma

```
./programa
```

o valor de *argc* é igual a 1, pois só existe um parâmetro (o próprio nome do programa). Na execução

```
./programa arq.txt valor
```

o valor de argc é igual a 3, pois existem três parâmetros: programa, arq.txt e valor.

O outro argumento da função *main* é chamado de *argv* (do inglês: **arg**ument **v**ector). Ele é um vetor de *strings*, em que cada posição armazena um dos parâmetros passados ao programa (incluindo seu nome). No nosso primeiro exemplo, *argc* é igual a 1, e *argv[0]* contém “programa”. No segundo exemplo, *argc* é igual a 3 e *argv*, consequentemente, possui três posições: *argv[0]* que contém “programa”, *argv[1]* que contém “arq.txt” e *argv[2]* que armazena “valor”. Para mais informações a respeito de passagem de parâmetros via linha de comando para programas em C consulte um dos livros da bibliografia da disciplina ou o link abaixo:

http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node145.html

A dinâmica desse laboratório segue o estilo dos laboratórios anteriores.

Avaliação:

Uso de variável global: -5
Um vazamento de memória: -20%
Mais de um vazamento de memória: -40%
Não compila: 0.
Expirou o tempo de 15 segundos: 0.
Erro de execução (fim de programa anormal): 0.
Erro com ponteiro inválido: 0.
Uso da função exit: 0.
Obs: não utilize em seu arquivo .c caracteres especiais (á,é,õ,ã, etc) e comentários de múltiplas linhas iniciando e terminando em linhas diferentes.

Toda as operações do TDA numero_grande_t possuem o mesmo peso na avaliação.

Para comentar múltiplas linhas siga o exemplo abaixo:

```
//linha 1 de um comentario
//linha 2 de um comentario
//linha 3 de um comentario
```

ou

```
/*linha 1 de um comentario */
/*linha 2 de um comentario */
/*linha 3 de um comentario */
```