- **Detalhamento:** Este programa foi concebido para receber uma entrada de dados no seguinte formato: a primeira linha da entrada de dados contém as dimensões de uma matriz de inteiros; a segunda linha contém as coordenadas da entrada do labirinto (uma posição da matriz); a terceira linha contém as coordenadas da saída do labirinto (uma outra posição da matriz); as outras linhas contêm os elementos da matriz propriamente dita, constituída de elementos que podem ser 0 (zero) ou -1 (menos 1). Os zeros representam posições pelas quais se pode "passar" na matriz, enquanto que os -1 representam "paredes". Um exemplo de entrada de dados é:

```
6 12 (ler dimensões)
1 1  (ler_entrada)
6 12 (ler_saída)
 0  0  0 -1  0  0  0 -1  0 -1  0  0
 0 -1  0 -1  0 -1  0  0  0 -1  0 -1
 0  0  0  0  0 -1  0 -1 -1 -1  0 -1
-1  0 -1 -1  0 -1  0  0  0  0  0  0
 0  0  0  0  0  0  0 -1 -1  0 -1  0
 0 -1  0 -1 -1 -1  0  0 -1  0 -1  0
```

  Note que um labirito pode possuir várias possibilidades de entrada e saída, mas o programa deve encontrar um caminho mínimo entre as coordenadas indicadas no arquivo de entrada passando apenas por elementos da matriz propriamente dita.

  A ideia do algoritmo é marcar cada posição da matriz que não seja uma parede com valores inteiros que representam a distância desta posição para a saída. Para isto, ele inicia marcando a posição da saída com o valor 1 e toda outra posição (L,C) é marcada com o valor K desde que exista algum vizinho já marcado com valor (K-1). Os vizinhos válidos são: o da esquerda (L,C-1), o da direita (L,C+1), o da posição de cima (L-1,C) e o da posição de baixo (L+1,C), desde que estas coordenadas estejam dentro dos limites da matriz.

  Após esta etapa, a matriz acima fica como mostrado na figura seguinte:

```
19 18 17 -1 13 12 11 -1 13 -1  7  8
18 -1 16 -1 14 -1 10 11 12 -1  6 -1
17 16 15 14 13 -1  9 -1 -1 -1  5 -1
-1 15 -1 -1 12 -1  8  7  6  5  4  3
15 14 13 12 11 10  9 -1 -1  6 -1  2
16 -1 14 -1 -1 -1 10 11 -1  7 -1  1
```

- **Implementação:** Para implementar esta ideia, o programa principal, contendo ainda uma linha em pseudo-código, usa o seguinte algoritmo (que depende da correta inicialização das estruturas de dados): marca a posição da saída com o valor 1 (um) e insere seus vizinhos válidos no final de uma fila. Em seguida, até que a fila esvazie completamente, retira uma coordenada do início da fila, descobre qual é a distância K que esta tem da saída e para cada um dos seus vizinhos válidos: (1) anota $K + 1$ no vizinho; e (2) insere o vizinho no final da fila.

- **Menor caminho:** Na última linha do programa, após o laço que marca as distâncias na matriz, existe uma chamada a um procedimento que recebe uma matriz preenchida como descrito acima e encontra o menor caminho para se chegar da entrada do labirinto na sua saída. O menor caminho é encontrado partindo-se da entrada do labirinto e escolhendo sucessivamente em cada etapa o vizinho que tem um valor menor que o atual, até se chegar na saída, que tem o valor 1. Basta anotar de alguma maneira este caminho e em seguida imprimí-lo.

- **Questão única (100 pontos):** Sem fazer este último procedimento, sua tarefa é refinar o programa principal e implementar todas as outras funções e procedimentos.

- **Questão bônus (10 pontos extra):** Da maneira como você achar conveniente, implemente o procedimento que imprime o menor caminho.