MAC0323 - Algoritmos e Estruturas de Dados II

nome: Sabrina Araújo da Silva
nº USP: 12566182
professor: Carlos Eduardo Ferreira

Implementação:

- na função `main()` o algoritmo recebe os valores de m x n que compõem as dimensões do tabuleiro, que é iniciado com `char tabuleiro[m][MAX];`.
  - MAX é uma variável que foi criada para auxiliar na passagem da matriz do tabuleiro como parâmetro para algumas funções, já que em C++ é preciso que ao menos uma das dimensões seja declarada para usar a matriz como parâmetro.
- a função `pentaminos()` recebe o número de peças e o próprio tabuleiro.
  - é iniciado um array `int encaixadas[npecas];` que guarda quais peças já foram encaixadas no tabuleiro.
  - enquanto existir uma peça para ser encaixada:
    - o algoritmo percorre todas as 12 peças e procura uma que não foi encaixada para ver se é possível encaixar utilizando a função `encaixa()`. Essa função procura uma posição válida no tabuleiro e para cada uma das rotações da peça chama a função `rotacao()` que verifica se aquela rotação encaixa na posição válida, se de fato encaixar a função `pecaTab()` é chamada para atualizar o tabuleiro. Se isso ocorrer, o algoritmo procura novamente uma peça para encaixar.
    - se não foi encontrada nenhuma peça para encaixar e o número de encaixadas é menor que 12 o backtrack é iniciado. Caso a pilha não esteja vazia a última peça é desencaixada e se houver alguma rotação disponível para essa peça as tentativas de encaixar iniciam novamente a partir dessa rotação. Caso a pilha esteja vazia, significa que não é

possível preencher o tabuleiro e o algoritmo termina.

- a função `letra()` retorna a letra de cada peça como char.
- todas as peças foram armazenadas em uma matriz de 4 dimensões `const char pecas[12][9][5][5]`
  - a primeira guarda as 12 peças
  - a segunda tem todas as rotações de uma peça
  - a terceira e a quarta representam uma matriz 5 x 5 para guardar exatamente a própria variação da peça.
- a classe `stack` implementada contém 4 operações
  - `void push(int item)`: insere um elemento
  - `int pop()`: devolve o elemento do topo e o remove
  - `bool empty()`: verifica se está vazia
  - `int top()`: devolve o elemento do topo

  e foram criados um objeto stack para a pilha de peças e outro para uma pilha auxiliar de rotações.
- para iniciar o programa é preciso inserir os valores de m e n da matriz Amxn e em seguida o valor de cada uma das posições da matriz.

Exemplos de testes

matriz 3 x 20

```
00000000000000000000
00000000000000000000
00000000000000000000
```

input:

```
Insira os valores de m e n para criar o tabuleiro m x n:
3
20
Insira os valores da matriz 3 x 20
0
0
0
 ...
```

output:

```
Possivel!
UUXIIIIIZWWTTTFLLLLV
UXXXPPZZZYWWTFFFNNLV
UUXPPPZYYYYWTFNNNVVV
```

matriz 4 x 15

```
000000000000000
000000000000000
000000000000000
000000000000000
```

output:

```
Possivel!
FFNNNIIIIIZLLLL
VFFXNNWTZZZLUUU
VFXXXWWTZPPPUYU
VVVXWWTTTPPYYYY
```

matriz 5 x 12

```
000000000000
000000000000
000000000000
000000000000
000000000000
```

output:

```
Possivel!
FFVVVIIIIINN
LFFWVZZXNNNT
LFWWVZXXXTTT
LWWYZZUXUPPT
LLYYYYUUUPPP
```

matriz 6 x 10

```
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
```

output:

```
Possivel!
FFTTTWWVVV
IFFTWWXZZV
IFNTWXXXZV
INNPUUXYZZ
INPPUYYYYL
INPPUULLLL
```

matriz 8 x 8 com espaços vazios

```
00000000
00000000
00000000
00011000
00011000
00000000
00000000
00000000
```

output:

```
Possivel!
FFVVVUUU
LFFWVUXU
LFWWVXXX
LWW**NXI
LLZ**NNI
ZZZPPTNI
ZYPPPTNI
YYYYTTTI
```

matriz 5 x 13 com espaço vazio

```
0000000000000
0000000000000
0000000110000
0000001110000
0000000000000
```

output:

```
Possivel!
IVVVUUFFWWXLL
IVZZUFFWWXXXL
IVPZUUFW**XTL
IPPZZY***NNTL
IPPYYYYNNNTTT
```