Trabalho 2 – ED1

Robô x Labirinto

Entrega: 17/05
Trabalho em dupla
Entregar no Classroom um arquivo com códigos fonte (*.c, *.h) nomeado seguindo o padrão NomeSobrenome1_NomeSobrenome2_trab2.zip (ou .tgz)

O objetivo deste trabalho é aplicar o conceito de **pilha e fila** para que um fictício robô ande para todos os espaços possíveis de um labirinto (especificado através de um arquivo) retirando os prêmios no caminho.

Vocês podem implementar as TADs pilha e fila da forma que acharem mais interessante para esta aplicação, isto é, pode ser estática ou dinâmica (com ou sem nó cabeça, circular ou duplamente encadeada). Mas lembrem-se que as operações realizadas nessas estruturas são restritas.

Durante o trajeto, o robô deverá coletar prêmios e desviar de possíveis buracos. Ao final, ele deve imprimir na tela as coordenadas (linha x coluna) de cada prêmio na ordem em que foram encontrados, quantos ele encontrou e comparar com a quantidade total de prêmios existente no labirinto.

printf(“(%d,%d) ”, …); //imprimir linha e coluna de cada premio encontrado
printf(“\nPremios encontrados: %d\n”, ...); //quantidade de premios encontrados
printf(“Total de premios no labirinto: %d\n”, …); //quantidade total de premios no labirinto

O robô pode andar somente em quatro direções e obrigatoriamente na seguinte ordem de tentativa de direção: esquerda, direita, cima, baixo.

Como deve ser construído o labirinto:

- Deve ser armazenado em um arquivo do tipo texto com o nome “labirinto.txt”.
- A primeira linha deste arquivo deve conter a palavra “T2ROBO”;
- A segunda linha do arquivo é composta por dois números que indicam o número de linhas e de colunas do labirinto, respectivamente, incluindo a borda. É proibido o uso de alocação estática, devendo o espaço utilizado para armazenar o labirinto em memória ser alocado em tempo de execução (ou seja deve-se usar alocação dinâmica de memória para matrizes).
- Após as duas primeiras linhas do arquivo, o labirinto começa a ser definido.
    - O labirinto é basicamente constituído de **vazios** (indicados pelo caractere espaço ‘ ’), **paredes** (indicados pelo caractere ‘x’), **buracos** (indicados pelo caractere ‘b’), **prêmios** (indicados pelo caractere ‘p’) e uma **origem** (indicada pelo caractere ‘o’).

- Para evitar o tratamento de possíveis saídas do robô do labirinto, todas as “bordas” do labirinto devem ser preenchidas com paredes (‘x’).

- Para simplificação, todas as letras usadas para definir o labirinto devem estar em caixa baixa – minúscula.

Cada posição do labirinto deve ser implementada como um TAD, contendo dois campos: o tipo do campo; e um campo lógico para visita (visitado ou não). O labirinto deve ser implementado com um TAD composto por uma matriz de posições. A borda é considerada como parte da contagem do número de linhas e colunas. O caractere da esquerda superior fica na posição (0,0).

```
typedef struct posicao {
     char tipo;
     int visitado;
} Posicao;

typedef struct labirinto {
     Posicao **p;
} Labirinto;
```

Deve-se implementar as bibliotecas de pilha e fila em arquivos .h separados (pilha.h e fila.h).

Exemplo:

Entrada (arquivo):

```
T2ROBO
10 30
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x   x         b   x xp    x px
x x x xxxx xxxxxx x  x xx x xx
x x   x px x       x x x  x  x
xpx x xxxx   xxx p x   xxxxxpx
x xxx x xxxxx x    xpx x   x x
x x x x xx   x x       x xxx x
x xbx p  x x x   xxxx xb   x x
x   x xxox x   x x  p   xx   x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Saída:

```
(7,6) (4,1) (4,17) (5,20) (1,21) (8,20)
Premios encontrados: 6
Total de premios  no labirinto: 9
```