UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
ESTRUTURA DE DADOS
PROF. ME. ANDREIA DE JESUS

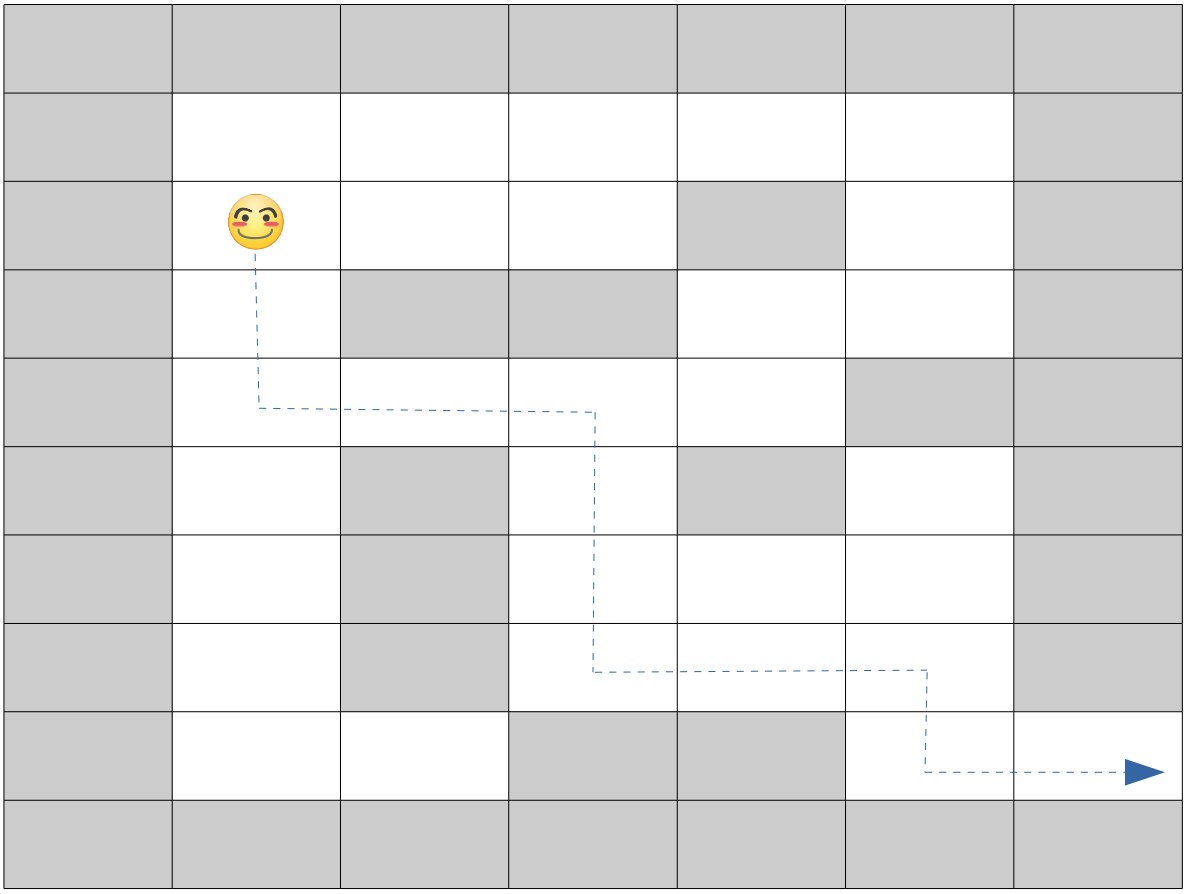
Postagem do Trabalho no Moodle – 15/05/2016
Defesas do Trabalho: 17 e 19/05/2016

1º TRABALHO PRÁTICO – ETAPA 2

1. **Aplicação de Pilha: O problema do labirinto (5.0 pontos)**

O objetivo deste exercício é usar uma pilha para implementar uma técnica conhecida como *backtracking* (ou retrocesso), frequentemente usada em Inteligência Artificial para resolver problemas por meio de tentativa e erro. Essa técnica é útil em situações em que, a cada instante, temos várias opções possíveis e não sabemos avaliar o melhor. Então, escolhemos uma delas e, caso essa escolha não leve à solução do problema, retrocedemos e fazemos uma nova escolha.

Para ilustrar o uso dessa técnica, aplicaremos o problema do rato em um labirinto. O rato está preso no labirinto e precisa achar o caminho que o live a à saída, conforme ilustra a imagem abaixo



Para tanto, você precisa:

- Declarar uma matriz quadrada 30X30 para representar o labirinto.
- As posições da matriz podem armazenar uma das seguintes marcas: livre, parede, visitada ou beco (você pode definir essas marcas como constantes inteiras). Inicialmente, toda posição é marcada como livre ou parede, e apenas posições livres podem ser alcançadas pelo rato .Quando uma posição livre é alcançada, sua marca é alterada para visitada e quando fica determinado que uma posição visitada conduz a um beco, sua marca é alterada para beco.
- Toda vez que um labirinto é criado, as bordas da matriz são marcadas como paredes e sua configuração interna é definida aleatoriamente. Além disso, a posição inicial do rato e a posição de saída do labirinto são marcadas como livres.
- Para definir a configuração interna da matriz, deve-se usar a função *random*.
- Para facilitar a visualização do processo de busca da saída do labirinto, ao exibir a matriz em vídeo, as posições devem ser marcadas como:
    - livre: será representada por um espaço em branco;
    - parede: será representada por um bloco sólido (ASCII 219);
    - visitada: será representada por um ponto;
    - beco: será representada por um bloco pontilhado (ASCII 176);
    - posição em que o rato se encontra no labirinto, no momento em que ele é exibido: será representada pelo caracter - ASCII 1 (carinha feliz)

  Obervação: A equipe pode escolher outra representação para os elementos do labirinto.
- Para encontrar a saída do labirinto, aplicar o seguinte algoritmo:
    - Definir (i,j) como posição corrente do rato, que inicialmente é (2,2);
    - Iniciar uma pilha P vazia;
    - Até que a posição corrente (i,j) se torne a posição de saída (n-1, n):
        - Marcar a posição corrente (i,j) como visitada;
        - Se houver uma posição livre adjacente a posição corrente, empilhamos a posição corrente e movimentamos o rato para essa posição livre;
        - Senão, estamos num beco e precisamos retroceder à última posição pela qual passamos para explorar outro caminho. Para isso, desempilhamos uma posição de P, que passa a ser a nova posição corrente. Caso a pilha esteja vazia, o labirinto não tem saída e a busca fracassa.
    - Alcançada a posição de saída a busca termina com sucesso.
- Para facilitar a manipulação da pilha, deve ser aplicado uma pilha de inteiros. Para tanto, deve ser transformado o par de coordenadas (i,j) num inteiro correspondente (**i * 100 + j)**. Por exemplo, o par de coordenadas (13,12) é empilhado como 13 * 100 + 12 que é igual a

1312. Ao desempilhar esse número, podemos restaurar o par de coordenadas original, fazendo **i = 1312 div 100** cujo resultado é 13 e **j = 1312 mod 100** cujo resultado é 12. Esse artifício funciona corretamente apenas quando cada coordenada tem no máximo dois dígitos.

Será avaliado os seguintes itens na implementação da solução desse problema:

- Manipulação de uma pilha encadeada não sequencial;
- Aplicação de funções, passagem de parâmetro por valor e por referência, retorno de função;
- Implementação de todos os elementos especificados acima

2. Aplicação de Fila: Fila de aviões para decolagem e pouso (3.0 pontos)

Faça um programa em C para simular um controlador de voo de um aeroporto. Neste programa o usuário deve ser capaz de realizar as seguintes tarefas:

- Listar o número de aviões esperando para decolar;
- Listar o número de aviões esperando para pousar;
- Autorizar a decolagem do primeiro avião na fila de decolagem;
- Autorizar o pouso do primeiro avião na fila de pouso;
- Adicionar um avião na fila de espera para decolagem;
- Adicionar um avião na fila de espera para pouso;
- Listar todos os aviões que estão na fila de espera para decolagem, por ordem de chegada;
- Listar todos os aviões que estão na fila de espera para pouso, por ordem de chegada;
- Listar as características do primeiro avião da fila, antes de autorizar a sua decolagem;
- Listar as características do primeiro avião da fila, antes de autorizar o seu pouso.

Considere que uma estrutura de dados do tipo fila encadeada seja usada para manipular os dados e que cada avião possui um nome, um identificador, uma origem e um destino. Se quiser coloque mais informações, número de passageiros, capacidade, modelo, etc. Para facilidade de implementação considere que existam duas pistas, uma só para decolagem e outra só para pouso.

Será avaliado os seguintes itens na implementação da solução desse problema:

- Manipulação de uma fila encadeada não sequencial;
- Aplicação de funções, passagem de parâmetro por valor e por referência, retorno de função;

- Implementação de todos os elementos especificados acima;
- Criatividade e originalidade na simulação dos pousos e decolagens na tela do computador.

ATENÇÃO:

1. As equipes devem ser as mesmas da Etapa I do Trabalho I.
2. Cópias de trabalho, todas as equipes envolvidas receberão nota zero.
3. **O prazo máximo para sanar dúvidas sobre este trabalho é: 12/05/2016**