Universidade do Estado do Amazonas - UEA
Escola Superior de Tecnologia - EST
Núcleo de Computação - NUCOMP

ALGORITMOS E ESTRUTURAS DE DADOS I
LISTA DE EXERCÍCIOS 4A – LISTAS COM IMPLEMENTAÇÃO ESTÁTICA
PROF. FLÁVIO JOSÉ MENDES COELHO

Listas com implementação estática simples

O brilhante professor Kaninchen está desenvolvendo um programa gerador de provas escolares (avaliações) para ajudar os professores na tarefa de confecção de provas. O programa funciona assim: (1) um professor põe um capacete com sensores de ondas mentais, e pensa na prova que deseja elaborar; (2) o programa recebe os sinais do professor via capacete e seleciona a partir de um banco de questões, aquelas que melhor avaliarão os alunos; (3) o programa copia as questões selecionadas para uma lista de questões; (4) o professor, então, pode verificar manualmente quais questões devem ficar, sair, e pode incluir outras questões na lista de questões. Como o projeto dos passos (1)-(3) foi orçado em milhões de dólares, o professor iniciará seu trabalho apenas com o passo (4). Ajude o tonificante professor Kaninchen a desenvolver seu programa. Para isto, codifique uma **lista** com implementação estática de acordo com os seguintes critérios:

1. Defina um registro Questão com os campos: id (chave, inteiro positivo), enunciado (descrição do enunciado da questão), popularidade (quantas vezes a questão já foi utilizada em provas passadas) e dificuldade (níveis de dificuldade: fácil, médio, bom, difícil, impossível).

2. A lista deverá ser implementada com vetor de questões de tamanho máximo e constante TAM (TAM $>$ 0).

3. Codifique a operação Cria(L) da lista. Esta operação inicializa a lista com tmamnaho zero.

4. Codifique a operação Mostra(L) da lista. Esta operação mostra na tela cada um dos itens de lista L.

5. Codifique a operação Insere(x, L) da lista. Esta operação inclui na lista L o item apontado pelo ponteiro x. Assuma que os atributos de x já estarão inicializados na rotina chamadora de Insere.

6. Codifique a operação Busca(k, L) da lista. Esta consulta retorna um ponteiro para o item em L, tal que $x.chave = k$, ou NIL se x não estiver em L.

7. Codifique a operação Remove(p, L) da lista. Esta operação de modificação rece uma posição p de um elemento x em L, e remove logicamente x de L (logicamente, significa que x não é fisicamente removido de L, mas sim, sobreposto por outro elemento).

8. Codifique a operação Minimo(L) da lista. Esta consulta sobre a lista ordenada L retornará um ponteiro para o elemento de L com a chave mínima.

9. Codifique a operação Máximo(L) da lista. Esta consulta sobre a lista ordenada L retornará um ponteiro para o elemento de L com a chave máxima.

10. Codifique a operação Sucessor(x, L) da lista. Dado um item x cuja chave está na lista ordenada L, retorna um ponteiro para o próximo maior item em L, ou NIL se x for o item máximo.

11. Codifique a operação Predecessor(x, L) da lista. Dado um item x cuja chave está na lista ordenada L, retorna um ponteiro para o próximo menor item em L, ou NIL se x for o item mínimo.

12. ★ Codifique a operação Busca-Dificuldade(d, L, D) da lista. Esta consulta inclui na lista D somente as questões da lista L com dificuldade igual ao valor de d. A lista D deve ser passada vazia para a operação.

13. ★ Codifique a operação Busca(k, L) da lista. Esta consulta emprega a estratégia de busca binária para retornar um ponteiro para o item em L, tal que $x.chave = k$, ou NIL se x não estiver em L.