Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Bacharelado em Engenharia de Automação e Controle

MATA40 - Estrutura de Dados | Período: 2015.2 | Data: 17/03/2016.
Prof. Antonio L. Apolinário Junior
Estagiário Docente: Alan Santos

Roteiro do Laboratório 3 - Listas Encadeadas com Apontadores

Objetivos:

- Reforçar os conceitos de alocação dinâmica de memória;
- Compreender de forma prática o conceito de encadeamento em listas baseadas em Apontadores;
- Implementar, em linguagem C, um TAD Lista Encadeada, baseada em apontadores;
- Estender a implementação para variantes das Listas Simplesmente Encadeadas, como Listas Duplamente Encadeadas e Listas Circulares.

Conceitos básicos:

Lista Encadeada baseada em Apontadores:

O uso de alocação dinâmica de memória (ver Lab. 2) pode ser extendido para que os nós da lista sejam alocados sob demanda, e não de forma arbitrária e em bloco como na solução baseada em Arranjos (ver Lab. 2). Dessa forma a estrutura de dados nó passa a apontar não para elemento do vetor (índice inteiro) mas para um endereço de memória que contem um outro nó. Ou seja:

```
typedef struct No {     float       dado; // informação armazenada no nó da lista
                        struct No*  prox; // encadeamento - endereço do nó sucessor
                } tNo;
```

Dessa forma a lista não necessita mais de um vetor/arranjo para suporte, mas apenas de um ponteiro para o primeiro nó da lista. Portanto, a **lista encadeada baseada em apontadores** é definida por:

```
typedef struct {   tNo *inicio; // armazena o endereço do primeiro nó da lista
                   int numElems; // numero atual de nos na lista
                   } tListaEncadeada;
```

Roteiro:

1. Baixe do repositório da disciplina () os códigos fonte base para esse Laboratório.

2. Analise a estrutura de arquivos que compõe esse Laboratório. Abra os arquivos e entenda onde esta o programa principal e quais são os módulos utilizados.

3. Compile os programas executando na linha do console o comando **`make`**.

4. Não havendo erros, execute o programa **`mainListaEncadeada`**. Verifique que, tal como no Laboratório anterior, as funções relativas ao TAD **`tListaEncadeada`** não estão todas implementadas, mas garantem um valor de retorno que permite que o programa principal seja executado.

5. Abra o arquivo **`listaEncadeada.c`** e analise as funções a serem implementadas.

6. Codifique a função **`initLista()`** para que ela, crie uma lista vazia.

7. Implemente na sequencia as funções **`tamLista()`** e **`imprimirLista()`**. Compile e teste suas rotinas usando o programa de **`mainListaEncadeada`**. A essa altura a execução do programa de teste deve ser capaz de imprimir as listas, ainda que vazias.

8. Codifique a função **`inserirElem()`** de modo que cada novo nó seja sempre inserido no final da lista. Compile e teste suas rotinas usando o programa de **`mainListaEncadeada`**;

9. A essa altura sua lista já deve estar sendo impressa com elementos gerados aleatoriamente. Implemente as funções **`buscaElem()`** e **`buscaElemPos()`**;

10. Implemente agora a função **`removerElem()`**.

11. Compile e teste suas rotinas usando o programa de **`mainListaEncadeada`**;

12. Crie mais duas funções de inserção de elementos na lista: uma que faz a inserção sempre no inicio da lista, e outra que insere um novo elemento de forma ordenada, presumindo que toda a lista naquele instante já se encontra ordenada. Teste suas novas funções alterando o programa **`mainListaEncadeada`** para criar duas novas listas onde as inserções são feitas apenas com as novas funções.

13. Avalie a possibilidade de armazenar na estrutura de dados da Lista Encadeada, também um ponteiro para o ultimo elemento da lista. Codifique essa variante do TAD **`tListaEncadeada`** e avalie as mudanças que cada uma de suas operações sofreriam.

Desafio do Final de Semana:

1) É possível conceber listas encadeadas em que se represente de forma explicita a relação de precedência entre nós. Nesse caso, os nós além de armazenarem o endereço do próximo nó também armazenam o endereço do nó anterior. Esse tipo de lista é chamado de **Lista Duplamente Encadeada**.

Defina e codifique esse novo TAD **`tListaDuplamenteEncadeada`** analisando as modificações na estrutura de dados e seus reflexos nas funções.

2) Outra variante interessante de Lista Encadeada é a **Lista Circular**. Como o nome sugere, esse tipo de lista forma um ciclo, ou seja, ao chegar no ultimo elemento este aponta de volta para o primeiro elemento.

   Defina e codifique esse outro TAD **`tListaCircular`** analisando as modificações na estrutura de dados e seus reflexos nas funções.

Para tornar esse desafio mais “emocionante” teremos uma tarefa aberta no **Moodle** para que vocês possam submeter sua versão do desafio até o final do domingo. Aqueles que conseguirem concorrerão a “prêmios” nas categorias: o mais rápido, a melhor solução, entre outras.