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: 03/03/2016.

Prof. Antonio L. Apolinário Junior

Estagiário Docente: Alan Santos

Roteiro do Laboratório 1 - Listas com Arranjos

Objetivos:

- Compreender o conceito de Tipo Abstrato de Dados (TAD);
- Reforçar o conceito de modularidade para implementação de TADs;
- Implementar, em linguagem C, um TAD Lista, baseado em arranjo.

Conceitos básicos:

Lista:

Seqüência de zero ou mais itens $\mathbf{x_1}$; $\mathbf{x_2}$; …. ; $\mathbf{x_n}$, na qual $\mathbf{x_i}$ é de um determinado tipo e **n** representa o tamanho da lista linear.

Sua principal propriedade estrutural envolve as posições relativas dos itens em uma dimensão. Assumindo **n >= 1**, $\mathbf{x_1}$ é o primeiro item da lista e $\mathbf{x_n}$ é o último item da lista:

1) $\mathbf{x_i}$ precede $\mathbf{x_{i+1}}$ para **i = 1; 2; ... ; n - 1**
2) $\mathbf{x_i}$ sucede $\mathbf{x_{i-1}}$ para **i = 2; 3; ... ; n**
3) o elemento $\mathbf{x_i}$ é dito estar na **i-ésima** posição da lista.

Roteiro:

1. Baixe do repositório da disciplina (http://homes.dcc.ufba.br/~apolinario/Disciplinas/2015.2/MATA40/Laboratorio%201%20-%20Listas%20com%20Arranjos/) os códigos fonte base para esse Laboratório.

2. Analise a definição do TAD **tListaVet** definido no arquivo **listaVetor.h**:

```
#define MAX_LISTA 50
typedef struct {  float V[MAX_LISTA];     // vetor que armazena os elementos da lista
                  int numElems;           // numero de elementos na lista em um
                                          // determinado momento
                  int tamMax;             // tamanho maximo da lista. <= MAX_LISTA
               } tListaVet;
```

3. Analise nesse mesmo arquivo as funções definidas para **tListaVet**. Nos comentários você vai encontrar a descrição, parâmetros e valor de retorno.

4. Abra o arquivo **listaVetor.c**. Verifique que as funções definidas em **listaVetor.h** serão codificadas nesse arquivo. E que todas as funções estão incompletas, sem parâmetros e código, apenas comandos return padrão, compatíveis com a definição da função.

5. Analise o arquivo **mainListas.c**, que contem o programa principal de uma das aplicações desse Laboratório. Verifique com atenção a sequencia de operações do TAD **listaVetor**.

6. Compile os programas executando na linha do console o comando make.

7. Não havendo erros, execute o programa **listaVetor**.

8. Volte ao arquivo **listaVetor.c** e implemente a função **initLista()** que permitirá que você crie uma lista vazia.

9. Compile novamente o programa e teste sua implementação.

10. Implemente na sequencia as funções **tamLista()**, **imprimirLista()** e **imprimirListaCompleta()**. Compile e teste suas rotinas usando o programa de **mainListas**. A essa altura a execução do programa de teste deve ser capaz de imprimir as listas

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

12. A essa altura sua lista já deve estar sendo impressa com elementos gerados aleatoriamente. Implemente agora a função **removerElem()**. Note que:

    12.1. para remover um elemento primeiro precisamos verificar se o elemento esta na lista;

    12.2. a remoção deve garantir que as propriedades 1), 2) e 3) de uma lista permaneçam válidas. Portanto, para garantir que essas propriedades sejam mantidas essa função deve reorganizar os elementos da lista.

13. Compile e teste suas rotinas usando o programa de **mainListas**;

14. Outra possibilidade de efetuar a remoção de um elemento de uma lista é fazer uma operação “lógica”, ou seja, apenas sinalizar na posição do vetor que ela agora é uma posição livre. Pense e reflita sobre essa possibilidade. Quais suas vantagens? E desvantagens?

15. Planeje e implemente uma nova versão da operação de **`removerElem()`** que faça a remoção de forma lógica. Cuidado pois essa mudança pode gerar a necessidade de alterar outras funções também.

16. Uma vez que as rotinas básicas do TAD tListaVetor estejam testadas e funcionando, abra o arquivo listaVetorInterativa.c. Nesse arquivo você vai encontrar uma nova aplicação que implementa um sistema interativo de manipulação de uma lista. Nele um menu de opções é definido para que o usuário selecione as principais funcionalidades de uma lista. Você deve agora acrescentar as funcionalidades da sua implementação das operações sobre uma tListaVetor, acionadas através dessa aplicação.

17. Teste e verifique se o funcionamento dos sistema esta correto.

Exercício de fixação:

1. Implemente funções mais sofisticadas para a sua lista:

   1.1. Fazer uma cópia da lista linear.

   1.2. Combinar duas ou mais listas lineares em uma lista única.

   1.3. Partir uma lista linear em duas ou mais listas.

2. Modifique os programas de teste fornecidos para que essas funções possam ser verificadas e testadas.