ESTRUTURAS DE DADOS - FUNDAMENTOS

Parte I

Abstração de Dados

Uma abstração é uma visualização ou uma representação de uma entidade que inclui somente os atributos de importância em um contexto particular [SEBESTA].

Um tipo abstrato de dados (TAD) é uma representação ou um modelo de um conjunto de dados, todos do mesmo tipo, e um conjunto de operações que manipulam os dados desse conjunto.

Poderíamos pensar em um modelo matemático, representado pelo par ordenado (E,O) onde E é o conjunto de elementos, todos do mesmo tipo, e O, as operações que manipulam esses elementos.

Exemplo: um conjunto de números inteiros e as operações (+, -, *, DIV, RESTO).

TAD Lista Linear

a) É uma seqüência de $n \geq 0$ elementos x_1, x_2, x_3,...,x_n, tal que:

- x_1 é o primeiro elemento da Lista.
- x_n é o último elemento da Lista.
- Para todo i, j, $1 \leq i, j \leq n$, se i <j então x_i é o antecessor de x_j.
- Para todo i, j, $1 \leq i, j \leq n$, se i >j então x_i é o sucessor de x_j.
- Se n=0, então a lista está vazia.
- n é a quantidade de elementos da lista.

b) As operações que permitem a manipulação desse conjunto: criar uma lista, inserir novo elemento na lista, remover um elemento da lista, buscar um elemento na lista, mostrar a lista, juntar listas, quebrar lista, ordenar seus elementos, etc.

Estrutura de Dados

Uma estrutura de Dados é uma implementação de um tipo abstrato de dados.

Podemos pensar em um Tipo Abstrato de Dados (TAD) como sendo um modelo conceitual e uma Estrutura de Dados (ED) com um modelo lógico. Um modelo lógico, então, depende da utilização de alguma linguagem de programação que permita implementar esse TAD.

Implementando uma Lista Linear

Existem pelo menos duas formas de implementar uma lista linear:

1. A primeira delas, mais comum, é através da utilização de arranjos (arrays unidimensionais). Esta forma de alocar memória é chamada de seqüencial ou estática.
2. A outra é através de alocação ligada ou dinâmica, utilizando variáveis do tipo ponteiro (apontadores), discutidos mais adiante.

Implementação de uma Lista Linear em alocação sequencial

Uma lista é utilizada para gerenciar, dentre várias aplicações, a memória RAM. Os itens são armazenados em posições contíguas de memória. Neste caso a lista pode ser percorrida em qualquer direção. Assim, não existe qualquer critério para a realização das operações na lista. A título de exemplo, suponha que a lista linear está armazenada em um vetor de tamanho MAX, de números inteiros. A figura, abaixo, mostra a parte do vetor ocupada pela Lista.

	Elementos								
números da lista	**9**	**3**	**8**	**1**	**6**	**7**	**2**	**...**	**...**
posição (índice)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**...**	**MAX-1**

Observe que **índice** indica a ultima posição. Para um novo elemento a ser inserido na lista, o valor de **índice** será incrementado em uma unidade **índice** = **índice** +1). Uma remoção fará que o valor de **índice** seja decrementado em uma unidade (**índice** = **índice** - 1).

Operações:

- **Inicializar:** Faremos o valor de **índice** = -1, indicando uma lista vazia.
- **Inserir:** Faremos **índice** = **índice** + 1 e, em seguida Elementos[**índice**] = x, onde **x** representa um novo valor.
- **Remover:** Faremos **índice** = **índice** - 1. Note que não teremos mais a referência da posição.
- **Lista vazia:** Se **índice** =-1, a Lista está vazia.
- **Lista cheia:** se **índice** = MAX-1, a Lista esta cheia.

Implementação utilizando C/C++

```
#ifndef LISTALINEAR_H_INCLUDED
#define LISTALINEAR_H_INCLUDED
#define MAX 10

template <typename Tipo>
struct ListaLinear{
      Tipo elementos[MAX];
      int indice;
   public:
      ListaLinear();
      void insere(Tipo x);
      Tipo remover();
      bool listacheia();
      bool listavazia();

};

template <typename Tipo>
ListaLinear<Tipo>::ListaLinear(){
       indice = -1;
}
template <typename Tipo>
void ListaLinear<Tipo>::insere(Tipo x){
       indice++;
       elementos[indice]=x;
}

template <typename Tipo>
Tipo ListaLinear<Tipo>::remover(){
       Tipo temp=elementos[indice];
       indice--;
       return temp;
}
```

A aplicação

```
#include <iostream>
#include "ListaLinear.h"

using namespace std;
int main(){
   cout << "Lista Linear" << endl;
   int opc; //opcao a selecionar
   int valor; //valor de entrada
   ListaLinear<int> lista; //a Lista Linear

   do{
      cout<<"1-insere"<<endl;
      cout<<"2-remove"<<endl;
      cout<<"3-quantidade de elemantos"<<endl;
      cout<<"4-mostra elementos"<<endl;
      cout<<"5-fim"<<endl;
      cout<<"Selecione:";
      cin>>opc;

      switch(opc)
      {
         case 1:
            cout<<"digite o valor para inserir:";
            cin>>valor;
            if(lista.listacheia())
               cout<<"lista cheia..."<<endl;
            else
               lista.insere(valor);
            break;
         case 2:
            if(lista.listavazia())
               cout<<"lista vazia...."<<endl;
            else
```

```
template <typename Tipo>
bool ListaLinear<Tipo>::listacheia(){
      return indice == MAX-1;
}
template <typename Tipo>
bool ListaLinear<Tipo>::listavazia(){
      return indice==-1;
}

#endif // LISTALINEAR_H_INCLUDED
```

```
                  cout<<"removido:
"<<lista.remover()<<endl;
                  break;
               case 3:
                  if(lista.listavazia())
                     cout<<"lista vazia..."<<endl;
                  else
                     cout<<"Qtde de
elementos:"<<lista.indice+1<<endl;
                  break;
               case 4:
                  cout<<" Elementos da Lista:"<<endl;
                  if(lista.listavazia())
                     cout<<"lista vazia..."<<endl;
                  else{
                     for(int i=0;i<=lista.indice;i++)
                        cout<<lista.elementos[i]<<" ";
                     cout<<endl;
                  }
                  break;
               case 5:
                  cout<<"fim...."<<endl;
                  break;
               default:
                  cout<<"opcao invalida"<<endl;
                  break;
            }

         }while(opc!=5);
         return 0;
      }
```

Exercícios:

1. Dada uma lista seqüencial ordenada L1, escreva os procedimentos que:
    a. Verifique se L1 está ordenada ou não (a ordem pode ser crescente ou decrescente).
    b. Faça uma cópia da lista L1 em outra lista L2;
    c. Remova os elementos de L1 colocando-os em L2.