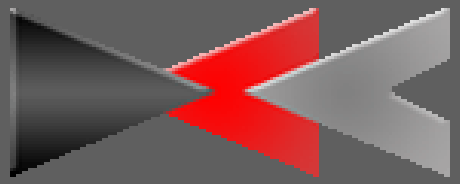
DCC107



Laboratório de Programação II

Tipos Abstratos de Dados Lista Contígua (TAD Lista Contígua) em C

Listas Lineares

- Uma lista é uma estrutura linear, composta de um conjunto de $n \geq 0$ elementos x_1, x_2, ..., x_n chamados **nós**, organizados de forma a manter a relação entre eles;
- Existem várias maneiras de representar uma lista, devendo ser escolhida a de melhor desempenho para a aplicação em questão;
- As representações mais comuns são por:
  - **Contiguidade** dos nós;
  - Encadeamento dos nós.

Nesta Aula

- TAD Lista:
  - Representação de lista por **contiguidade** dos nós;

TAD Lista

tipo LISTA

    domínio: LISTA, VALOR, ÍNDICE;

    operações:

        cria_lista(M) → LISTA;

        libera_lista(LISTA);

        acessa_no(LISTA, VALOR) → ÍNDICE;

        insere_no(LISTA, VALOR, ÍNDICE) → LISTA;

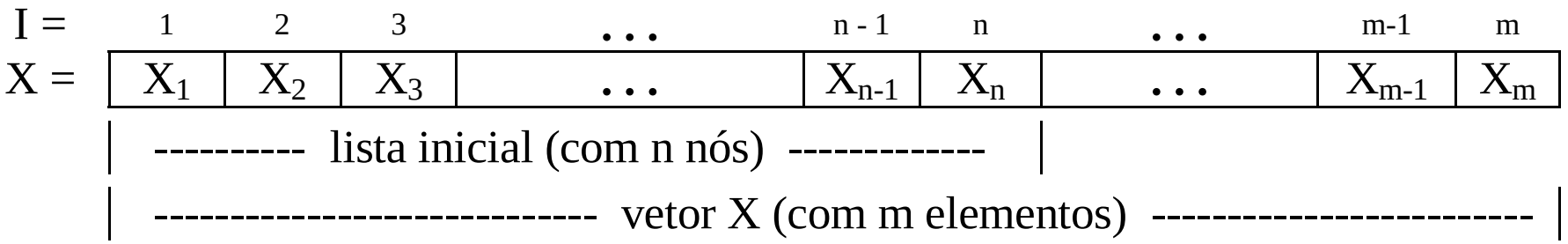
        elimina_no(LISTA, VALOR) → LISTA;

    fim-operações;

fim-tipo.

TAD Lista – Contiguidade

- Representação por **contiguidade** dos nós:
  - Também chamada de alocação sequencial;
  - Esquematicamente:



TAD Lista – Contiguidade

- Estrutura para representar o TAD Lista Contígua:

```
struct ListaCont
{
    int m, n; //m: capacidade maxima da lista
              //n: tamanho (numero de nos) da lista
    int *x;   //vetor de inteiros (info armazenada na lista)
};
```

  - Nesta estrutura há o vetor **x** onde são armazenadas as informações em cada nó (inteiro neste caso). Esta lista tem uma capacidade máxima **m** e o número de nós **n** da lista.

TAD Lista – Contiguidade

- Arquivos criados:
  - `ListaContigua.h`
  - `ListaContigua.c`
- Tipo criado:
  - `typedef struct ListaCont TlistaCont;`
- Operações implementadas:
  - `TlistaCont* criaListaVazia(int capacidade);`
  - `void liberaLista(TlistaCont *l);`
  - `void insere_extremo(TlistaCont *l, int valor);`
  - `void exibe_lista(TlistaCont *l);`
  - `int acessa_no(TlistaCont *l, int x);`
  - `void elimina_no(TlistaCont *l, int x);`

TAD Lista – Contiguidade

- Arquivo **`ListaContigua.h`**:

```
#ifndef LISTACONTIGUA_H_INCLUDED
#define LISTACONTIGUA_H_INCLUDED

//TAD Lista
typedef struct ListaCont TlistaCont;

//cria uma lista com 0 elementos com uma capacidade
TlistaCont* criaListaVazia(int capacidade);

//libera a memória ocupada pela lista
void liberaLista(TlistaCont *l);

//inserir um novo nó no extremo da lista
void insere_extremo(TlistaCont *l, int valor);
```

TAD Lista – Contiguidade

- Arquivo **`ListaContigua.h`** (continuação):

```
//inserir um novo nó no extremo da lista
void exibe_lista(TlistaCont *l);

//Acessa o nó Xk. Retorna -1 se Xk não está na lista
//ou o indice de Xk se está na lista
int acessa_no(TlistaCont *l, int x);

//Elimina o nó Xk
void elimina_no(TlistaCont *l, int x);

#endif // LISTACONTIGUA_H_INCLUDED
```

TAD Lista – Contiguidade

- Arquivo **`ListaContigua.c`**:

```
#include <stdio.h>
#include <stdlib.h>
#include "ListaContigua.h"

struct ListaCont
{
    int m, n; //m: capacidade máxima da lista
              //n: tamanho (número de nós) da lista
    int *x;   //vetor de inteiros (info armazenada na lista)
};
```

TAD Lista – Contiguidade

- Arquivo **`ListaContigua.c`** (continuação):

```
//cria uma lista com 0 elementos com uma capacidade
TlistaCont* criaListaVazia(int capacidade)
{
    TlistaCont* l = (TlistaCont*) malloc(sizeof(TlistaCont));
    l->x = (int*) malloc(sizeof(int)*capacidade);
    l->m = capacidade;
    l->n = 0;
    return l;
}
//libera memória ocupada pela lista
void liberaLista(TlistaCont *l)
{
    free(l->x);
    free(l);
}
```

TAD Lista – Contiguidade

- Arquivo **`ListaContigua.c`** (continuação):

```
//inserir um novo nó no extremo da lista
void insere_extremo(TlistaCont *l, int valor)
{
    if(l->n < l->m)
    {
        l->x[l->n] = valor;
        l->n++;
    }
    else
    {
        printf("A lista está cheia");
        exit(1);
    }
}
```

TAD Lista – Contiguidade

- Arquivo **`ListaContigua.c`** (continuação):

```
//exibe a lista
void exibe_lista(TlistaCont *l)
{
    int i;
    for(i = 0; i < l->n; i++)
        printf("\n%d",l->x[i]);
}
```

TAD Lista – Contiguidade

- Arquivo **`ListaContigua.c`** (continuação):

```
//Acessa o nó Xk. Retorna -1 se Xk não está na lista
//ou o indice de Xk se está na lista
int acessa_no(TlistaCont *l, int x)
{
    int k;
    for(k = 0; k < l->n; k++)
        if(l->x[k] == x)
            return k; //achou
    return -1; //não achou
}
```

TAD Lista – Contiguidade

- Arquivo **`ListaContigua.c`** (continuação):

```
//Elimina o nó Xk
void elimina_no(TlistaCont *l, int x)
{
    int i, k = acessa_no(l, x);
    if(k != -1)
    {   //desloca os elementos com índice >= k para a esquerda
        for(i = k; i < l->n-1; i++)
            l->x[i] = l->x[i+1];
        l->n--;
    }
}
```

TAD Lista – Contiguidade

- Arquivo **`main.c`**:

```
#include <stdio.h>
#include <stdlib.h>
#include "ListaContigua.h"

int main()
{
    int i;

    //cria uma lista vazia com capacidade de 10 inteiros
    TlistaCont *l = criaListaVazia(10);

    //insere 10 números inteiros na lista l (4, 5, ..., 13)
    for(i = 4; i <= 13; i++)
        insere_extremo(l, i);
```

TAD Lista – Contiguidade

- Arquivo **`main.c`** (continuação):

```
    //exibe todo o conteúdo da lista
    exibe_lista(l);

    //exibe o índice cujo conteúdo do nó é 8
    printf("\nValor 8: %d\n", acessa_no(l, 8));

    //elimina o nó cujo conteúdo é 8
    elimina_no(l, 8);
    exibe_lista(l);

    //libera a memória ocupada pela lista
    liberaLista(l);
    return 0;
}
```

Exercícios – Contiguidade

- Criar operações para:
  1. Inserir um nó na posição k;
  2. Inserir um nó antes do nó X_k (elemento na posição k);
  3. Inserir um nó depois do nó X_k (elemento na pos. k);
  4. Concatenar duas listas;
  5. Partir uma lista em duas a partir de um nó dado X_k;
  6. Ordenar uma lista, ou seja, colocá-la, por exemplo, com todos os nós em ordem crescente.
  7. Criar um TAD Lista Contígua de forma que todos os nós da lista ficam sempre ordenados.