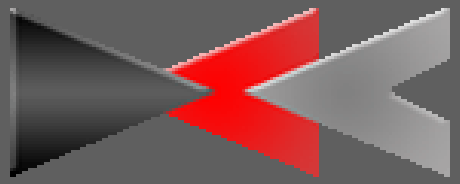
DCC107



Laboratório de Programação II

Tipos Abstratos de Dados Lista Encadeada (TAD Lista Encadeada) 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 **encadeamento** 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;

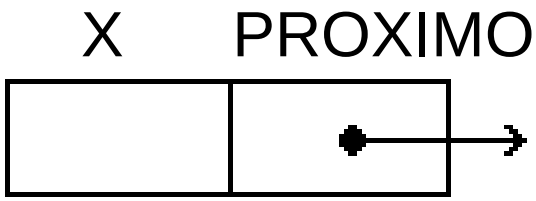
    fim-operações;

fim-tipo.

TAD Lista – Encadeamento

- Representação por **encadeamento** dos nós:
  - Lista simplesmente encadeada;
  - Esquematicamente:

  - Sendo um nó representado por:



TAD Lista – Encadeamento

- Estrutura para representar o TAD Lista Encadeada:

```
struct lista
{
    int info;
    struct lista *prox;
};
```

  - Nesta estrutura **`info`** representa a informação em cada nó e **`prox`** é um ponteiro para o próximo nó da lista.

TAD Lista – Encadeamento

- Arquivos criados:
  - `ListaEncadeada.h`
  - `ListaEncadeada.c`
- Tipo criado:
  - `typedef struct lista Lista;`
- Operações implementadas:
  - `Lista* inicializa();`
  - `Lista* insere(Lista *l, int i);`
  - `void imprime(Lista *l);`
  - `int vazia(Lista *l);`
  - `Lista* busca(Lista *l, int v);`
  - `Lista* retira(Lista *l, int v);`
  - `void libera(Lista *l);`

TAD Lista – Encadeamento

- Arquivo **`ListaEncadeada.h`**:

```
#ifndef LISTAENCADEADA_H_INCLUDED
#define LISTAENCADEADA_H_INCLUDED

/* função de inicialização: retorna uma lista vazia */
Lista* inicializa();

/* inserção no início: retorna a lista atualizada */
Lista* insere(Lista *l, int i);

/* função imprime: imprime valores dos elementos */
void imprime(Lista *l);
```

TAD Lista – Encadeamento

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

```
/* função vazia: retorna 1 se vazia ou 0 se não vazia */
int vazia(Lista *l);

/* função busca: busca um elemento na lista */
Lista* busca(Lista *l, int v);

/* função retira: retira elemento da lista */
Lista* retira(Lista *l, int v);

/* função libera: libera a memória ocupada pela lista */
void libera(Lista *l);

#endif // LISTACONTIGUA_H_INCLUDED
```

TAD Lista – Encadeamento

- Arquivo **`ListaEncadeada.c`**:

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

struct lista
{
    int info;
    struct lista *prox;
};
/* função de inicialização: retorna uma lista vazia */
Lista* inicializa()
{
    return NULL;
}
```

TAD Lista – Encadeamento

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

```
/* inserção no início: retorna a lista atualizada */
Lista* insere(Lista *l, int i)
{
    Lista *novo = (Lista*) malloc(sizeof(Lista));
    novo->info = i;
    novo->prox = l;
    return novo;
}
/* função imprime: imprime valores dos elementos */
void imprime(Lista *l)
{
    Lista *p; /* variável auxiliar para percorrer a lista */
    for(p = l; p != NULL; p = p->prox)
        printf("info = %d\n", p->info);
}
```

TAD Lista – Encadeamento

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

```
/* função vazia: retorna 1 se vazia ou 0 se não vazia */
int vazia(Lista *l)
{
    return (l == NULL);
}
/* função busca: busca um elemento na lista */
Lista* busca(Lista *l, int v)
{
    Lista *p;
    for(p = l; p != NULL; p = p->prox)
        if(p->info == v)
            return p;
    return NULL; /* não achou o elemento */
}
```

TAD Lista – Encadeamento

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

```
/* função retira: retira elemento da lista */
Lista* retira(Lista *l, int v)
{
    Lista *ant = NULL; /* ponteiro para elemento anterior */
    Lista *p = l; /* ponteiro para percorrer a lista*/
    /* procura elemento na lista, guardando anterior */
    while(p != NULL && p->info != v)
    {
        ant = p;
        p = p->prox;
    }
    /* verifica se achou elemento */
    if(p == NULL)
        return l; /* não achou: retorna lista original */
```

TAD Lista – Encadeamento

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

```
    /* retira elemento */
    if(ant == NULL)
    {
        /* retira elemento do inicio */
        l = p->prox;
    }
    else
    {
        /* retira elemento do meio da lista */
        ant->prox = p->prox;
    }
    free(p);
    return l;
}
```

TAD Lista – Encadeamento

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

```
/* função libera: libera a memória ocupada pela lista */
void libera(Lista *l)
{
    Lista *p = l;
    while(p != NULL)
    {
        Lista *t = p->prox;/*guarda referência p/ próximo elemento*/
        free(p); /* libera a memória apontada por p */
        p = t; /* faz p apontar para o próximo */
    }
}
```

TAD Lista – Encadeamento

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

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

int main()
{
    Lista *l = inicializa(); // declara e inicia um lista vazia
    l = insere(l, 23); // insere na lista o elemento 23
    l = insere(l, 45); // insere na lista o elemento 45
    l = insere(l, 56); // insere na lista o elemento 56
    l = insere(l, 78); // insere na lista o elemento 78

    printf("\n\n");
    imprime(l); // imprimirá 78 56 45 23
```

TAD Lista – Encadeamento

- Arquivo **`main.c`** (continuação):

```
    l = retira(l, 78);
    printf("\n\n");
    imprime(l); // imprimirá 56 45 23

    l = retira(l, 45);
    printf("\n\n");
    imprime(l); // imprimirá 56 23

    libera(l);

    return 0;
}
```

Exercícios – Encadeamento

1. Considerando uma lista simplesmente encadeada, criar operações para:
   a) Inserir um nó antes do nó X_k (elemento na posição k);
   b) Inserir um nó depois do nó X_k (elemento na pos. k);
   c) Concatenar duas listas;
   d) Determinar o número de nós de uma lista;
   e) Partir uma lista em duas a partir de um nó dado X_k;
   f) Ordenar uma lista, ou seja, colocá-la, por exemplo, com todos os nós em ordem crescente;
   g) Comparar duas listas (verificar se são iguais).
2. Criar um TAD Lista simplesmente encadeada de forma que todos os nós da lista ficam sempre ordenados.

Exercícios – Encadeamento

3. Criar um TAD Lista:
   a) Com descritor;
   b) Circular;
   c) Duplamente encadeada;
   d) Circular duplamente encadeada.

- Considerando as seguintes operações:
  - inicializa;
  - insere;
  - imprime;
  - vazia;
  - busca;
  - retira;
  - libera.