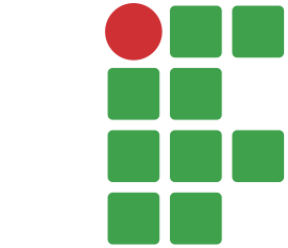
LISTAS ENCADEADAS



INSTITUTO FEDERAL
Paraíba

Professor Msc Paulo de Tarso F. Júnior

paulodt@gmail.com

Agenda

- Motivação
- Listas encadeadas
- Listas circulares
- Listas duplamente encadeadas
- Listas de tipos estruturados

Motivação

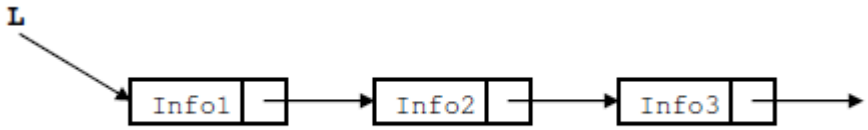
- Vetor
  - Ocupa um espaço contíguo de memória
  - Permite acesso randômico aos elementos
  - Deve ser dimensionado com um número máximo de elementos

Motivação

- Estruturas de dados dinâmicas:
  - Crescem (ou decrescem) à medida que elementos são inseridos (ou removidos)
  - Exemplo:
    - Listas encadeadas:
      - Amplamente usadas para implementar outras estruturas de dados

Lista Encadeada

- Lista encadeada:
  - Sequência encadeada de elementos, chamados de *nós da lista*
  - Nó da lista é representado por dois campos:
    - Informação armazenada
    - Ponteiro para o próximo elemento da lista
  - A lista é representada por um ponteiro para o primeiro nó
  - O ponteiro do último elemento é NULL



Lista Encadeada

- Exemplo:
  - Lista encadeada armazenando valores inteiros
  - Estrutura lista
    - Estrutura dos nós da lista
  - Tipo Lista
    - Tipo dos nós da lista

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

lista é uma estrutura auto-referenciada, pois o campo prox é um ponteiro para uma próxima estrutura do mesmo tipo uma lista encadeada é representada pelo ponteiro para seu primeiro elemento, do tipo Lista*

A palavra reservada typedef nada mais é do que um atalho em C para que possamos nos referir a um determinado tipo existente com nomes sinônimos.

Lista Encadeada

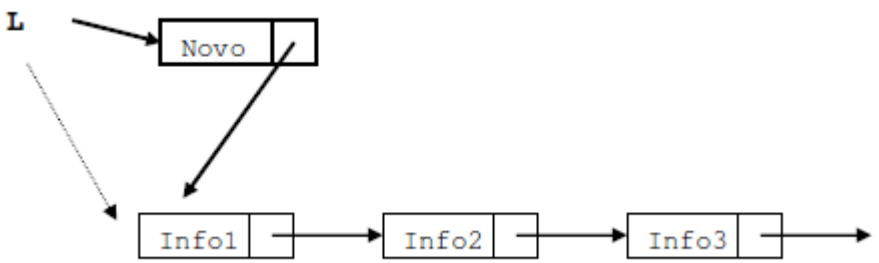
- Exemplo - Função de criação
  - Cria uma lista vazia, representada pelo ponteiro NULL

```
/* função de criação: retorna uma lista vazia */
Lista* criaLista (void)
{
      return NULL;
}
```

Lista Encadeada

- Exemplo - Função de inserção
  - Aloca memória para armazenar o elemento
  - Encadeia o elemento na lista existente

```
/* inserção no início: retorna a lista atualizada */
Lista* insereLista (Lista* lista, int i)
{
      Lista* novo = (Lista*) malloc(sizeof(Lista));
      novo->info = i;
      novo->prox = lista;
      return novo;
}
```



Lista Encadeada

- Exemplo - Trecho de código
  - Cria uma lista inicialmente vazia e insere novos elementos

```
int main (void)
{
      Lista* lista; /* declara uma lista não inicializada */
      lista = criaLista(); /* cria e inicializa lista como vazia */
      lista = insereLista(lista, 23); /* insere na lista o elemento 23 */
      lista = insereLista(lista, 45); /* insere na lista o elemento 45 */
      return 0;
}
      ...
```

deve-se atualizar a variável que representa a lista
a cada inserção de um novo elemento.

Lista Encadeada

- Exemplo - Função para imprimir uma lista
  - Imprime os valores dos elementos armazenados

```
/* função imprime: imprime valores dos elementos */
void imprimeLista (Lista* l)
{
        Lista* p;
        for (p = l; p != NULL; p = p->prox){
                printf("info = %d\n", p->info);
        }
}
```

variável auxiliar p:

- **ponteiro, usado para armazenar** o endereço de cada elemento
- **dentro do loop, aponta para cada** um dos elementos da lista

Lista Encadeada

- Exemplo - Função para verificar se uma lista está vazia
  - Retorna 1 se a lista estiver vazia ou 0 se não estiver vazia

```
/* função vazia: retorna 1 se vazia ou 0 se não vazia */
int listaVazia (Lista* l)
{
	return (l == NULL);
}
```

Lista Encadeada

- Exemplo - Função de busca
  - Recebe a informação referente ao elemento a pesquisar
  - Retorna o ponteiro do nó da lista que representa o elemento, ou NULL, caso o elemento não seja encontrado na lista

```
/* função busca: busca um elemento na lista */
Lista* buscaLista (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 */
}
```

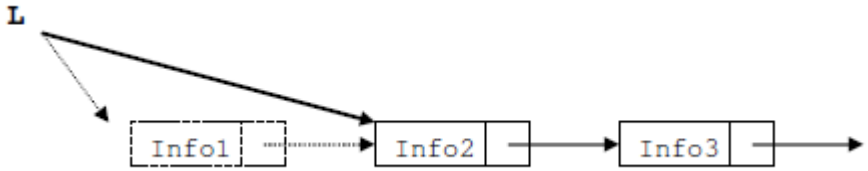
Lista Encadeada

- Exemplo - Função para liberar a lista
  - Destrói a lista, liberando todos os elementos alocados

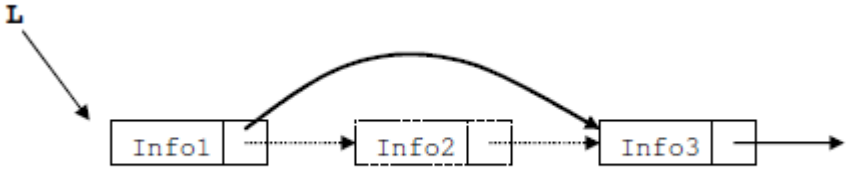
```
void liberaLista (Lista* l){
	Lista* p = l;
	while (p != NULL) {
		Lista* t = p->prox; /* guarda referência p/ próx. elemento */
		free(p); /* libera a memória apontada por p */
		p = t; /* faz p apontar para o próximo */
	}
}
```

Lista Encadeada

- Exemplo - Função para retirar um elemento da lista
  - Recebe como entrada a lista e o valor do elemento a retirar
  - Atualiza o valor da lista, se o elemento removido for o primeiro



  - Caso contrário, apenas remove o elemento da lista



Lista Encadeada

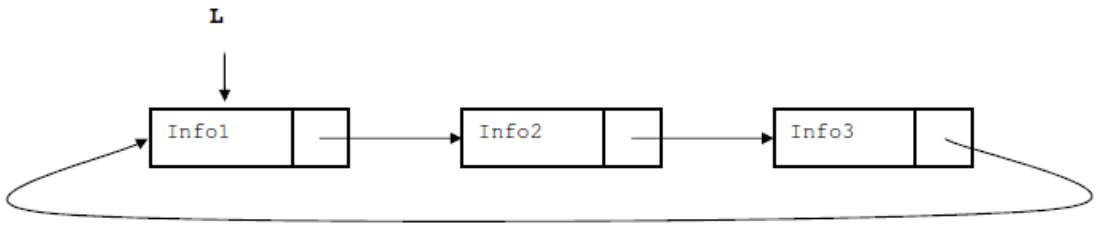
```
/* função retira: retira elemento da lista */
Lista* retiraLista (Lista* l, int v)
{
	Lista* ant = NULL; /* ponteiro para elemento anterior */
	Lista* p = l; /* ponteiro para percorrer a lista */

	/* 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 */

	/* 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;
}
```

Lista Encadeada - Utilização

```
#include <stdio.h>
#include "lista.h"
int main (void) {
      Lista* l; /* declara uma lista não iniciada */
      l = criaLista(); /* inicia lista vazia */
      l = insereLista(l, 23); /* insere na lista o elemento 23 */
      l = insereLista(l, 45); /* insere na lista o elemento 45 */
      l = insereLista(l, 56); /* insere na lista o elemento 56 */
      l = insereLista(l, 78); /* insere na lista o elemento 78 */
      imprimeLista(l); /* imprimirá: 78 56 45 23 */
      l = retiraLista(l, 78);
      imprimeLista(l); /* imprimirá: 56 45 23 */
      l = retiraLista(l, 45);
       imprimeLista(l); /* imprimirá: 56 23 */
      liberaLista(l);
      return 0;
}
```

Listas Circulares

- Lista circular:
  - O último elemento tem como próximo o primeiro elemento da lista, formando um ciclo
  - A lista pode ser representada por um ponteiro para um elemento inicial qualquer da lista



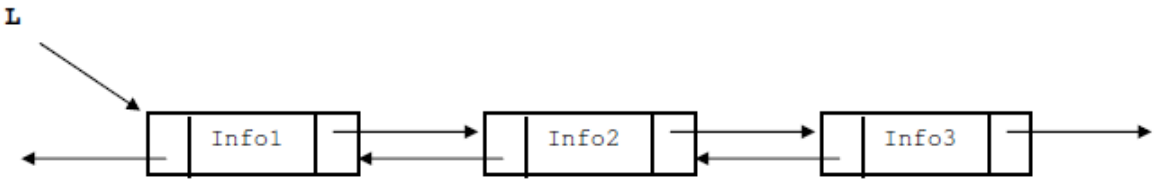
Listas Circulares

- Exemplo - Função para imprimir uma lista circular
  - Visita todos os elementos a partir do ponteiro do elemento inicial até alcançar novamente esse mesmo elemento
  - Se a lista é vazia, o ponteiro para um elemento inicial é NULL

```
/* função imprime: imprime valores dos elementos */
void imprimeListaCircular (Lista* l)
{
      Lista* p = l; /* faz p apontar para o nó inicial */
      /* testa se lista não é vazia e então percorre com do-while */
      if (p) do {
            printf("%d\n", p->info); /* imprime informação do nó */
            p = p->prox; /* avança para o próximo nó */
      } while (p != l);
}
```

Lista Duplamente Encadeada

- Lista duplamente encadeada:
  - Cada elemento tem um ponteiro para o próximo elemento e um ponteiro para o elemento anterior
  - Dado um elemento, é possível acessar o próximo e o anterior
  - Dado um ponteiro para o último elemento da lista, é possível percorrer a lista em ordem inversa



Lista Duplamente Encadeada

- Exemplo:
  - Lista encadeada armazenando valores inteiros
  - Estrutura lista2
    - Estrutura dos nós da lista
  - Tipo Lista2
    - Tipo dos nós da lista

```
struct lista2 {
        int info;
        struct lista2* ant;
        struct lista2* prox;
};
typedef struct lista2 Lista2;
```

Lista Duplamente Encadeada

- Funções:
  - Criação
  - Inserção
  - Impressão
  - Lista Vazia
  - Busca
  - Liberação
  - Remoção

Lista de Tipos Estruturados

- Lista de tipo estruturado:
  - A informação associada a cada nó de uma lista encadeada pode ser mais complexa, sem alterar o encadeamento dos elementos
  - As funções apresentadas para manipular listas de inteiros podem ser adaptadas para tratar listas de outros tipos

Lista de Tipos Estruturados

- Lista de tipo estruturado (cont.):
  - O campo da informação pode ser representado por um ponteiro para uma estrutura, em lugar da estrutura em si
  - Independente da informação armazenada na lista, a estrutura do nó é sempre composta por
    - Um ponteiro para a informação e
    - Um ponteiro para o próximo nó da lista

Lista de Tipos Estruturados

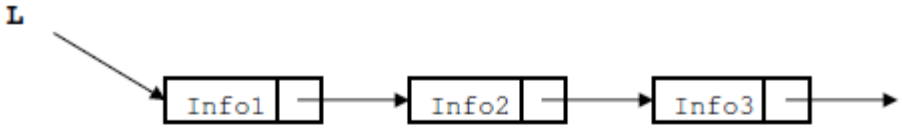
- Exemplo – Lista de retângulos

```
struct retangulo {
        float b;
        float h;
};
typedef struct retangulo Retangulo;

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

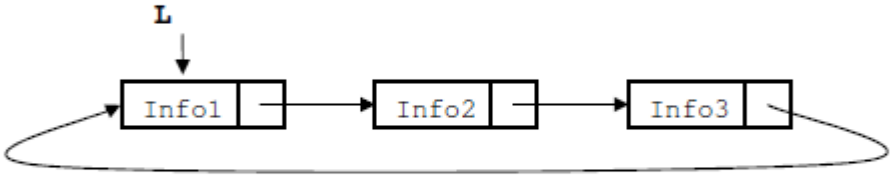
campo da informação representado por um ponteiro para uma estrutura, em lugar da estrutura em si

Resumo

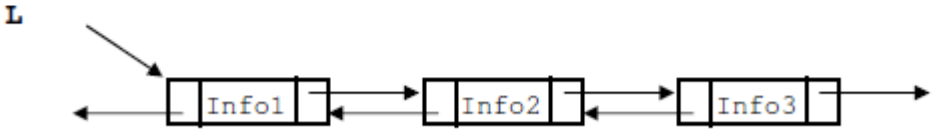
Listas Encadeadas



Listas Circulares



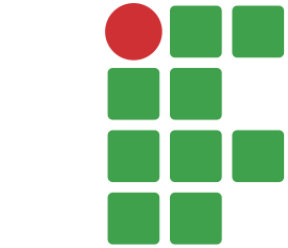
Listas duplamente encadeadas



Referências

- Waldemar Celes, Renato Cerqueira, José Lucas Rangel, Introdução a Estruturas de Dados, Editora Campus (2004) Capítulo 10 – Listas encadeadas
- **Tipos de Listas em C**. Disponível em: http://www.cprogressivo.net/2013/10/Lista-simplesmente-encadeada-com-cabeca-em-C-Inserindo-nos-no-inicio-e-fim.html. Acesso em 15/08/2017

LISTAS ENCADEADAS



INSTITUTO FEDERAL
Paraíba

Professor Msc Paulo de Tarso F. Júnior

paulodt@gmail.com