

INSTITUTO FEDERAL
BAIANO



Estrutura de Dados

Listas Encadeadas ou Listas Dinâmicas

Prof. José Honorato Ferreira Nunes

honoratonunes@softwarelivre.org

<http://softwarelivre.org/zenorato>

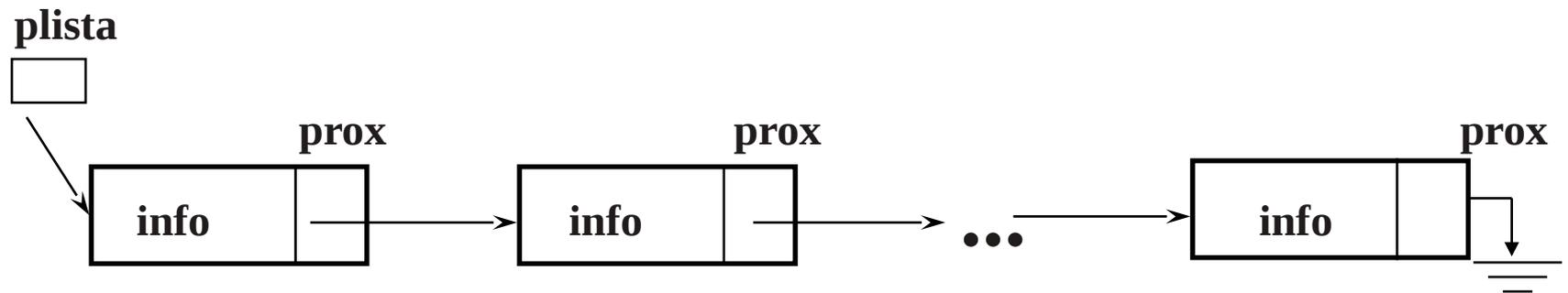
Listas Encadeadas - Conceitos Básicos

É uma sequências de Nós, cuja relação de sucessão é estabelecida por um ponteiro de endereço do próximo Nó.

Diferentemente das listas sequenciais que vimos até este momento, em que a sucessão dos Nós é estabelecida de forma física ou contínua, nas listas encadeadas, a sucessão dos Nós é estabelecida de forma lógica.

Listas Encadeadas Dinamicamente

- Esta estrutura é tida como uma sequência de elementos encadeados por ponteiros, ou seja, cada elemento deve conter, além do dado propriamente dito, uma referência para o próximo elemento da lista.



Comando typedef

O comando typedef é usado para criar “sinônimo” ou um “alias” para tipos de dados existentes. Então na prática podemos dizer que estamos renomeando um tipo de dados.

Sintaxe:

```
typedef <nome do tipo de dado> <novo nome>;
```

É importante ressaltar que o comando typedef não cria um novo tipo. Ele apenas permite que um tipo existente seja denominado de uma forma diferente, de acordo com a especificação desejada pelo programador.

Comando typedef

```
{  
    //redefinição do tipo float como nota  
    typedef float nota;  
  
    //declarando as variáveis usando o tipo de dados renomeado  
    nota prova1, prova2, media;  
  
    printf ("Informe a nota da primeira prova: ");  
    scanf ("%f", &prova1);  
    printf ("Informe a nota da segunda prova: ");  
    scanf ("%f", &prova2);  
  
    media = (prova1 + prova2) / 2;  
    printf ("Media do aluno: %.2f\n", media);  
}
```

Função malloc

A função malloc (o nome é uma abreviatura de memory allocation) aloca um bloco de bytes consecutivos na memória RAM (random access memory) do computador e devolve o endereço desse bloco.

O número de bytes é especificado no argumento da função.

No seguinte fragmento de código, malloc aloca 1 byte:

```
char *ptr;  
ptr = malloc (1);  
scanf ("%c", ptr);
```

Função malloc

O endereço devolvido por malloc é do tipo genérico void *. O programador armazena esse endereço num ponteiro de tipo apropriado.

Para alocar um objeto que ocupa mais de 1 byte, é preciso recorrer ao operador sizeof, que diz quantos bytes o tipo de objeto desejado tem:

```
typedef struct {  
    int dia, mes, ano;  
} data;  
data *d;  
d = malloc (sizeof (data));  
d->dia = 31; d->mes = 12; d->ano = 2016;
```

Declaração da Estrutura de Listas Encadeadas

A estrutura definida é auto-referenciada, pois, além do campo para armazenar a informação, há um campo que é um ponteiro para uma próxima estrutura do mesmo tipo.

O tipo *no* representa um Nó da lista, sendo que a variável do tipo ponteiro **ptr* é uma estrutura do tipo *node*.

```
typedef struct no{
    int dado;                /* campo da informação */
    struct no* prox;        /* campo do ponteiro para o próximo nó */
} tno;                      /* tipo do nó */
typedef tno* tlista;       /* tipo lista */
```

Criação de uma Lista

- Uma função que cria uma lista vazia, deve ter como valor de retorno uma lista sem nenhum elemento.
- Como a lista é representada pelo ponteiro para o primeiro elemento, uma lista vazia é representada pelo ponteiro *NULL*, pois não existem elementos na lista.
- A função tem como valor de retorno, a lista vazia inicializada, isto é, o valor de retorno é *NULL*.

```
void criar(tlista *L){  
    *L = NULL;  
}
```

Função que aloca um Nó

- ✓ A função *malloc()* da biblioteca C aloca uma porção da memória para ser utilizada de acordo com a definição da estrutura.

```
tno* aloca(){  
    tlista p;  
    p = (tlista) malloc(sizeof(tno));  
    return(p);  
}
```

Função que insere um Nó na Lista

```
int inserir(tlista *L, int valor){
    tlista novo;

    novo = aloca();
    if (novo == NULL) return 0;
    novo->dado = valor;
    novo->prox = *L;
    *L = novo;
    return 1;
}
```

Função que libera um Nó

Após a exclusão de um item da lista, a área da memória deve ser liberada.

```
void libera(ptr p) {  
    free(p);  
}
```

A rotina acima utiliza a função *free()*, que serve para liberar a área de memória correspondente a variável *p*.

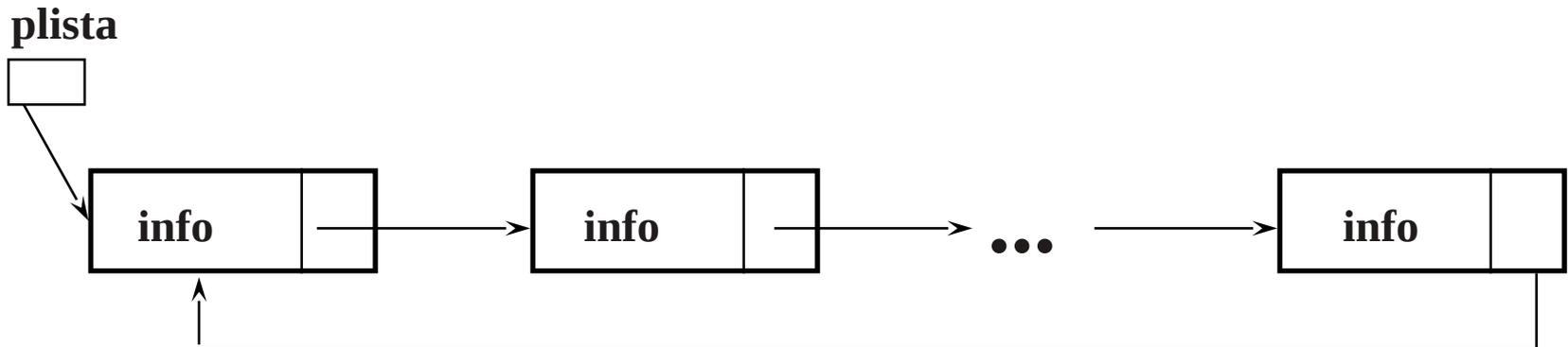
Função que remove um Nó na Lista

```
void remove(tlista *L){
    tlista del;
    if (*L == NULL){
        printf("Remoção nula");
        exit(1);
    }

    del = *L;
    *L = del->prox;
    free(del);
}
```

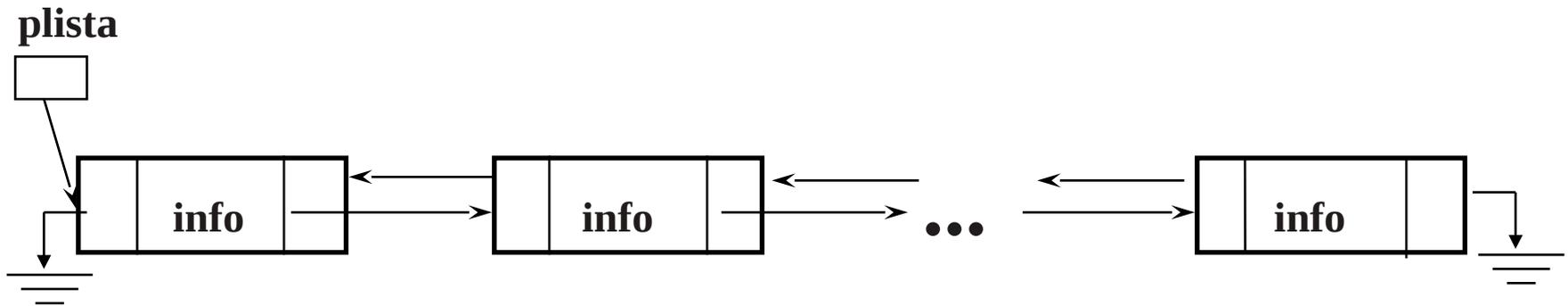
Lista Circular Simples (Encadeamento Circular)

Esta representação é uma modificação da lista simples onde o *último elemento aponta para o primeiro*.



Lista Duplamente Encadeada

Neste caso, cada nó da lista tem dois campos de apontadores: um apontador para seu predecessor (ou nó à esquerda) e outro para o seu sucessor (ou nó à direita).



Atividade

Faça um programa que, utilizando as funções criadas nos exemplos dessa aula crie uma fila F e exiba o seguinte menu de opções:

EDITOR DE LISTA

- 1 - INSERIR
- 2 - REMOVER
- 3 - EXIBIR A LISTA
- 4 - ESVAZIAR A LISTA
- 5 - CONTAR VALORES MAIORES QUE K
- 6 - SAIR

DIGITE SUA OPÇÃO:

Bibliografia Complementar

- ❑ BENEDUZZI, Humberto M. e METZ, João A. **Lógica e Linguagem de Programação - Introdução ao Desenvolvimento de Software** (1ª edição). Editora do Livro Técnico, 2010
- ❑ FORBELLONE, A. L. V. e Eberspacher, H. F. **Lógica de Programação - a Construção de Algoritmos e Estruturas de Dados** (3ª edição). Pearson, 2005
- ❑ CORMEN, Thomas H. et. al. **Algoritmos: Teoria e Prática**. Editora Campus, 2002.
- ❑ ZIVIANI, Nivio. **Projeto de Algoritmos**. Editora Nova Fronteira, 2004.
- ❑ SEBESTA, Robert W. **Conceitos de Linguagens de Programação**. Bookman, 2001.