

## Lista Dinâmica Encadeada

Prof. Bruno Travençolo  
(com adaptações feitas por Paulo H. R. Gabriel)

76

### Lista Dinâmica Encadeada

- ▶ Lista que utiliza alocação dinâmica (individual) para cada elemento
  - ▶ Ex: para cada aluno inserido, um malloc é feito.
- ▶ Se cada elemento é alocado dinamicamente, cada um retornará um endereço de memória indicando seu local de alocação
- ▶ Quantos endereços teremos?
  - ▶ Um para cada elemento
  - ▶ Lista com 1000 elementos, 1000 endereços
- ▶ Como lidar com esses endereços? Onde armazená-los?

▶

77

## Lista Dinâmica Encadeada

- ▶ Vetor de ponteiros?
  - ▶ Implicaria em gerenciar uma lista de ponteiros
- ▶ Solução
  - ▶ Incluir, para cada elemento da lista, o endereço do seu “vizinho”, ou seja, do próximo elemento da lista



78

## Exemplo: Lista de alunos

- ▶ Suponha que queremos guardar lista de alunos. As informações que queremos dos alunos está na estrutura *struct aluno*

```
struct aluno {
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
```

- ▶ Além do aluno, devemos armazenar o endereço do próximo aluno da lista

```
struct aluno *prox; // endereço do próximo aluno
```



79

## Listas encadeadas

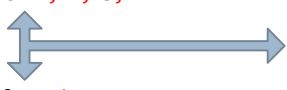
- ▶ A informação que queremos armazenar, juntamente com o endereço (ponteiro) pra próxima informação forma o que chamamos de **nó da lista**
- ▶ Outros nomes: *nó*; *node*; elemento
- ▶ Note que fazemos então um ponteiro para o próximo elemento **da lista** e não para o próximo dado

```

struct aluno {
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

struct lista_no {
    struct aluno dado;
    struct lista_no *prox;
};

```



80

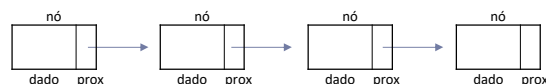
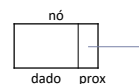
## Listas encadeadas

- ▶ Representação gráfica

```

struct lista_no {
    struct aluno dado;
    struct lista_no *prox;
};

```



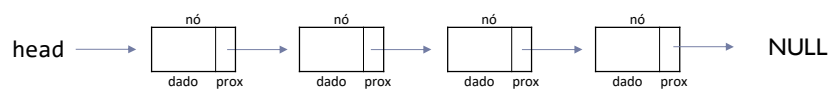
81

## Listas encadeadas

- ▶ Após a definição do nó, para definir a lista para indicar quem é o primeiro nó (nó cabeça – *head*).

```
struct lista {
    struct lista_no *head;
};
```

- ▶ O último nó deve ter como “próximo” o NULL, pois este é o final da lista



82

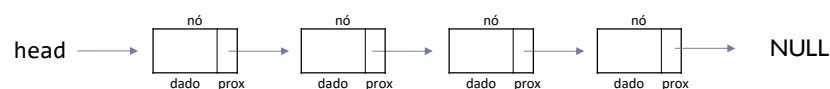
## Listas encadeadas

- ▶ Definição (com typedefs)

```
typedef struct lista Lista;
typedef struct lista_no Lista_no;

struct lista {
    Lista_no *head;
};

struct Lista_no {
    struct aluno dado;
    Lista_no *prox;
};
```



83

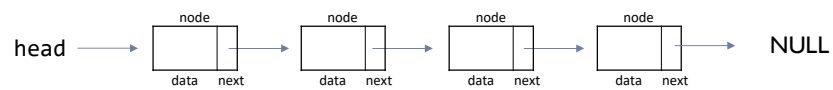
## In English

### ► Definição (com typedefs)

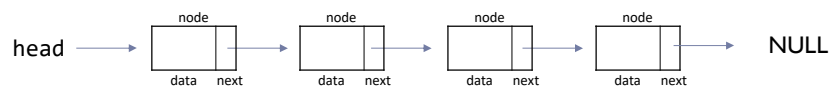
```
typedef struct list List;
typedef struct list_node List_node;

struct list {
    List_node *head;
};

struct List_node {
    struct aluno data;
    List_node *next;
};
```

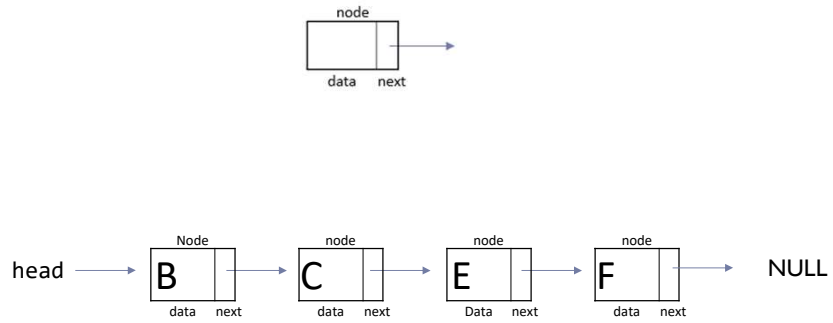


84



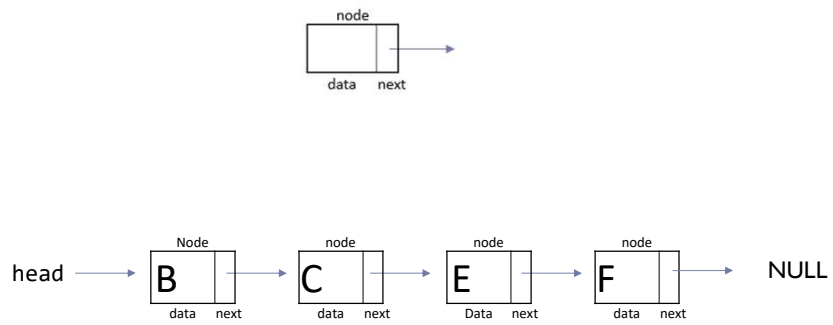
85

Insert D



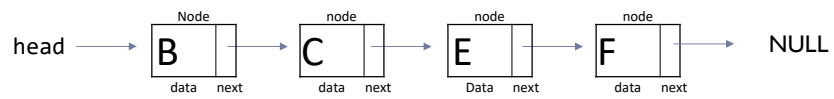
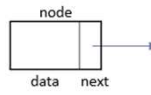
86

A



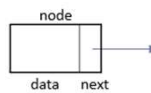
87

G ?



88

G ?



head → NULL

89

## Lista Dinâmica Encadeada

---

- ▶ Devido à alocação ser dinâmica para cada elemento os elementos da lista não ocuparão espaço contíguo
- ▶ Por não ser contíguo, não é possível saber de antemão em que local da memória foi alocado um determinado elemento da lista
  - ▶ Lembre-se: em vetores o espaço era contíguo, permitindo todos os elementos de um vetor sejam igualmente acessíveis

---

▶