

Listas Encadeada Simples

Prof. Denio Duarte

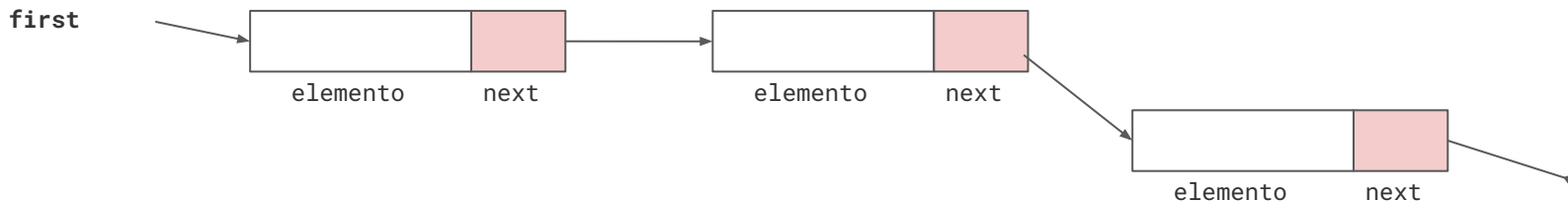
duarte@uffs.edu.br

Prof. Caio V. Koch

caio.santos@uffs.edu.br

Lista encadeada simples

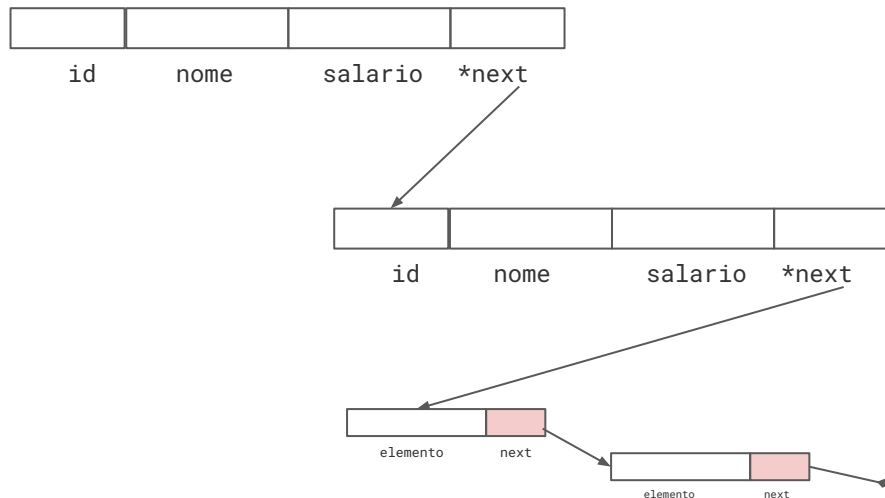
- Uma lista encadeada simples representa uma sequência de objetos, do mesmo tipo, na memória. Cada elemento da sequência armazena seu valor e o endereço do próximo elemento
 - Ou seja, junto a cada um dos elementos da lista, explicitamente armazenamos o endereço para o próximo elemento da lista



Lista encadeada

- Uma lista encadeada simples representa uma sequência de objetos, de mesmo tipo, na memória. Cada elemento da sequência armazena seu valor e o endereço do próximo elemento
 - Ou seja, junto a cada um dos elementos da lista, explicitamente armazenamos o endereço para o próximo elemento da lista

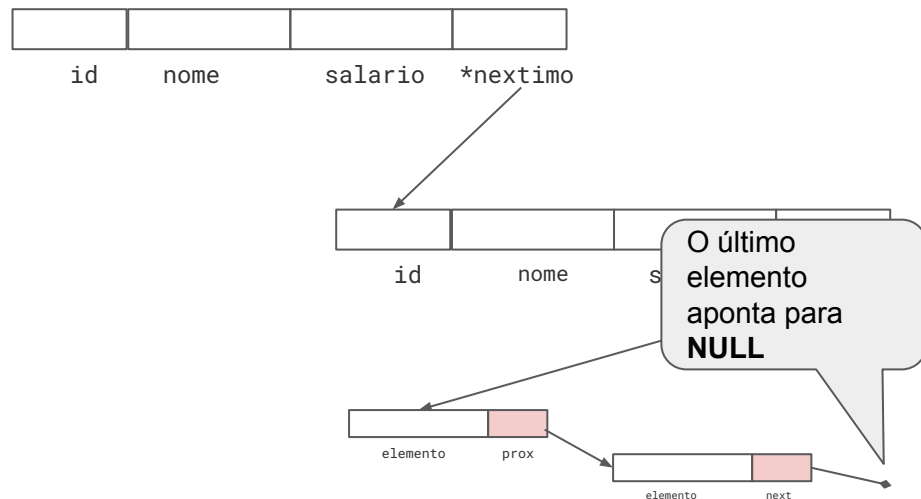
```
struct funcionario{  
    int id;  
    char nome[TAM_NOME+1];  
    double salario;  
    struct funcionario *next;  
};  
typedef struct funcionario Funcionario;
```



Lista encadeada

- Uma lista encadeada simples representa uma sequência de objetos, do mesmo tipo, na memória. Cada elemento da sequência armazena seu valor e o endereço do próximo elemento
 - Ou seja, junto a cada um dos elementos da lista, explicitamente armazenamos o endereço para o próximo elemento da lista

```
struct funcionario{  
    int id;  
    char nome[TAM_NOME+1];  
    double salario;  
    struct funcionario *next;  
};  
typedef struct funcionario Funcionario;
```



Lista encadeada

- Uma lista encadeada simples representa uma sequência de objetos, do mesmo tipo, na memória. Cada elemento da sequência armazena seu valor e o endereço do próximo elemento
- Os elementos de uma lista não ocupam uma área contígua de memória (como os vetores), o que não permite acesso direto aos elementos.
- Para acessar um elemento, é necessário que todos os elementos estejam encadeados.

```
struct funcionario{  
    int id;  
    char nome[TAM_NOME+1];  
    double salario;  
    struct funcionario *next;  
};  
typedef struct funcionario Funcionario;
```

Lista encadeada simples

- Como imprimimos os elementos
 - Para imprimir devemos iterar sobre todos os elementos partindo do first

```
struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;
```

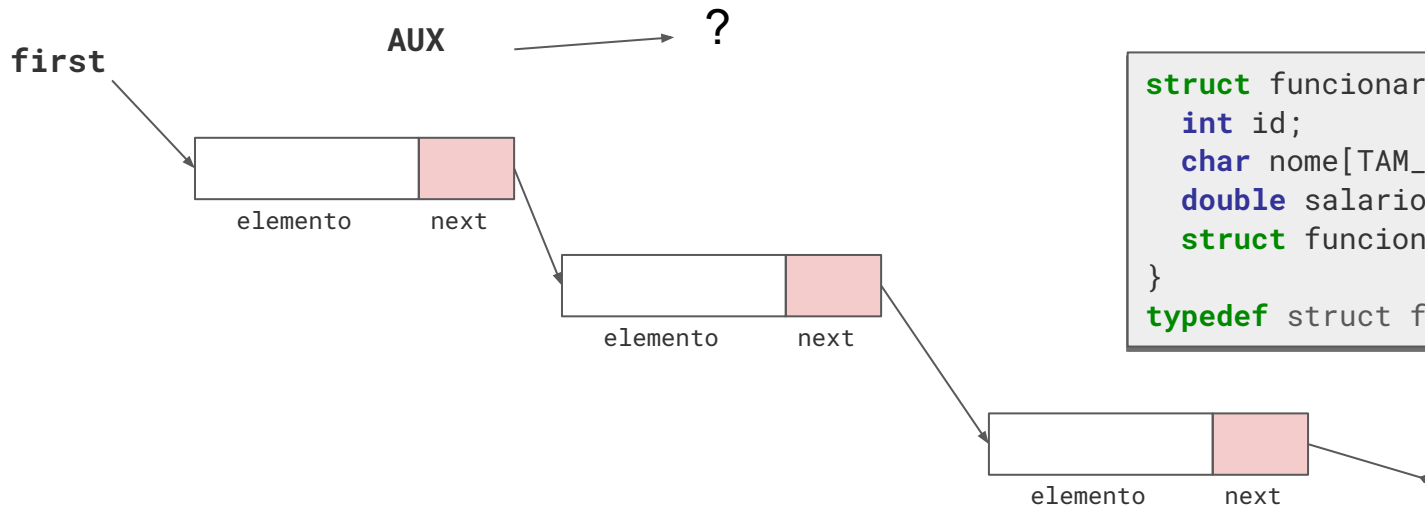
```
Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
}
```

```

Funcionario *aux; //vai
for (aux = first; aux != NULL; aux = aux->next) {
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
}

```

Iniciamos pelo first elemento



```

struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;

```

```

Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
}

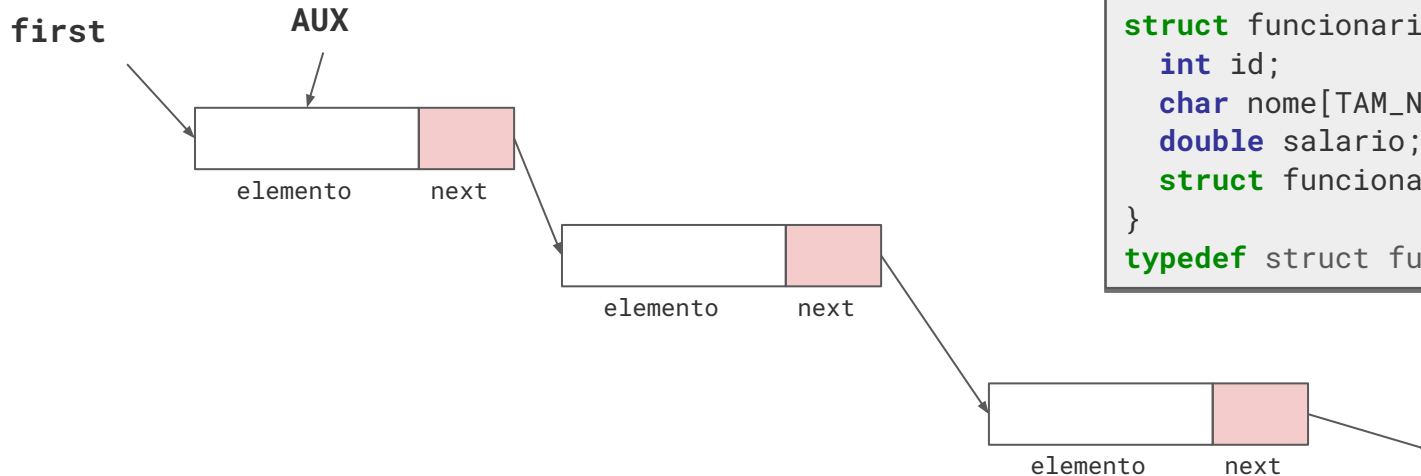
```

Imprime o conteúdo de AUX

```

printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);

```



```

struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;

```



```
Funcionario *aux; //vai ser nosso 'contador'
```

```
for (aux = first; aux != NULL; aux = aux->next){
```

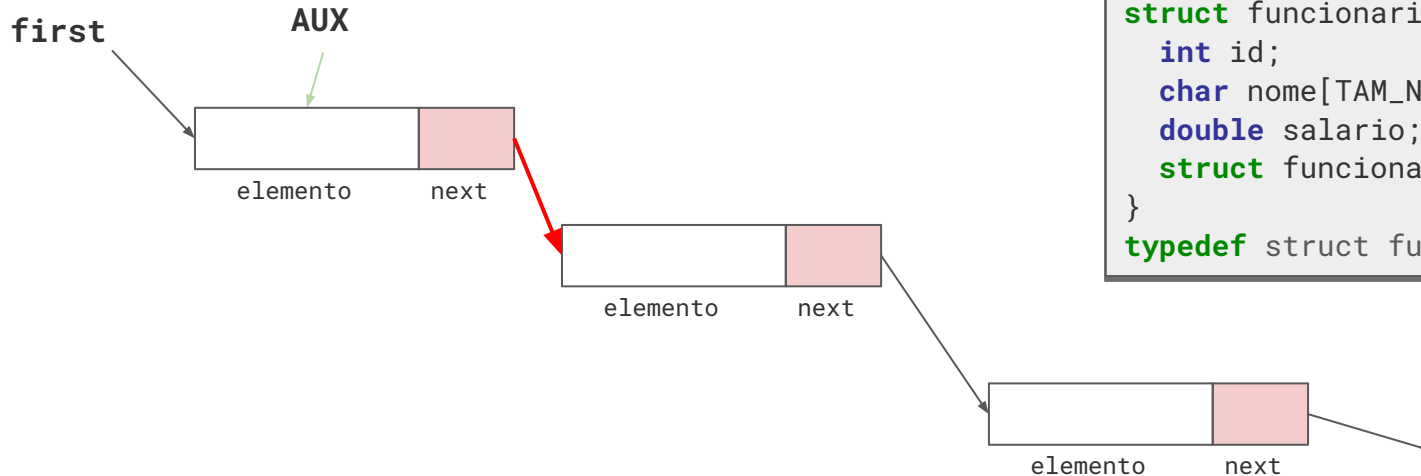
```
    //aqui aux vale o elemento atual na lista
```

```
    printf("Funcionario id: %d, nome: %s,
```

```
}
```

Faz o aux apontar para o próximo

```
->nome, aux->salario);
```



```
struct funcionario{  
    int id;  
    char nome[TAM_NOME+1];  
    double salario;  
    struct funcionario *next;  
}  
typedef struct funcionario Funcionario;
```

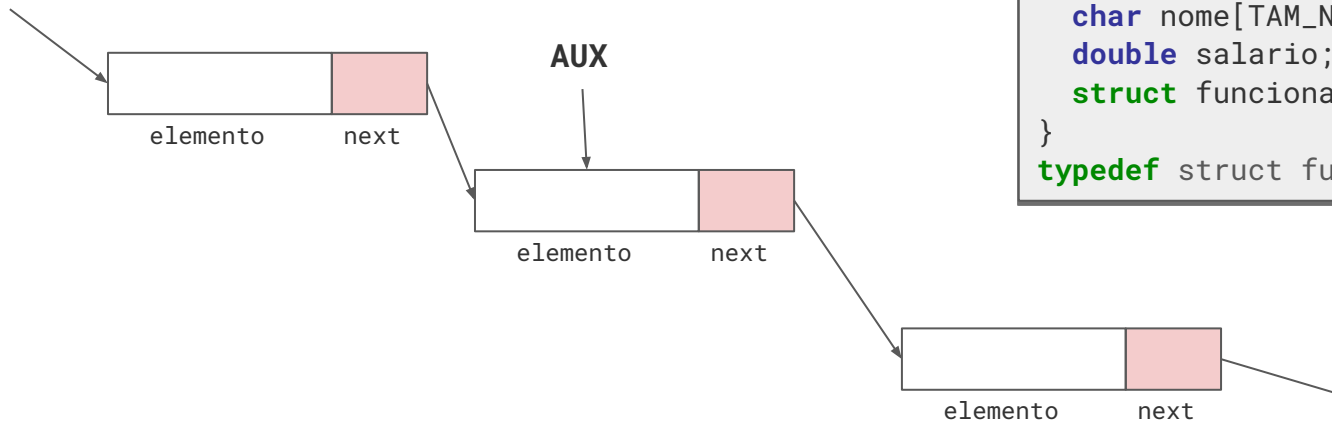
```

Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %f\n", aux->id, aux->nome, aux->salario);
}

```

Faz o aux apontar para o próximo

first



```

struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;

```

```

Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
}

```

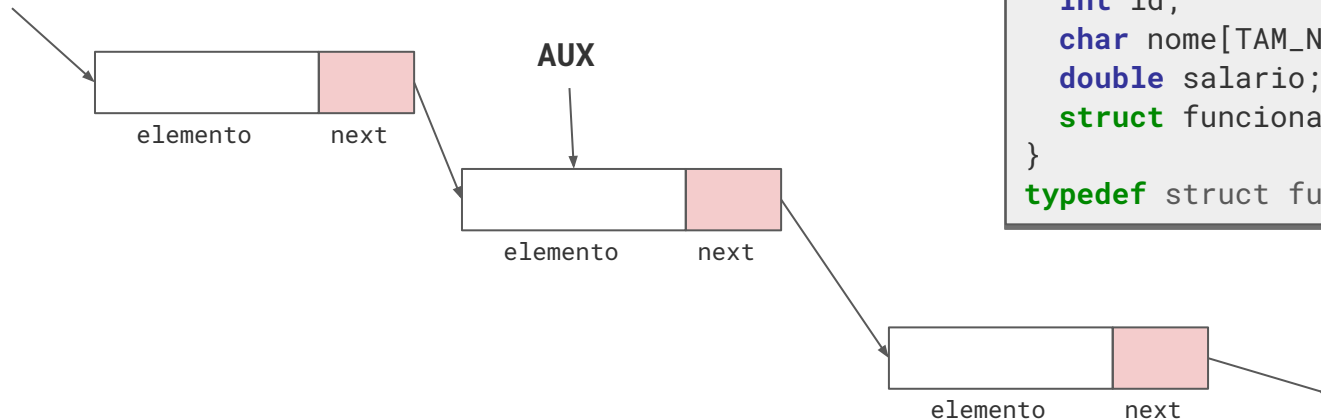
Imprime o conteúdo de AUX

```

printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);

```

first



```

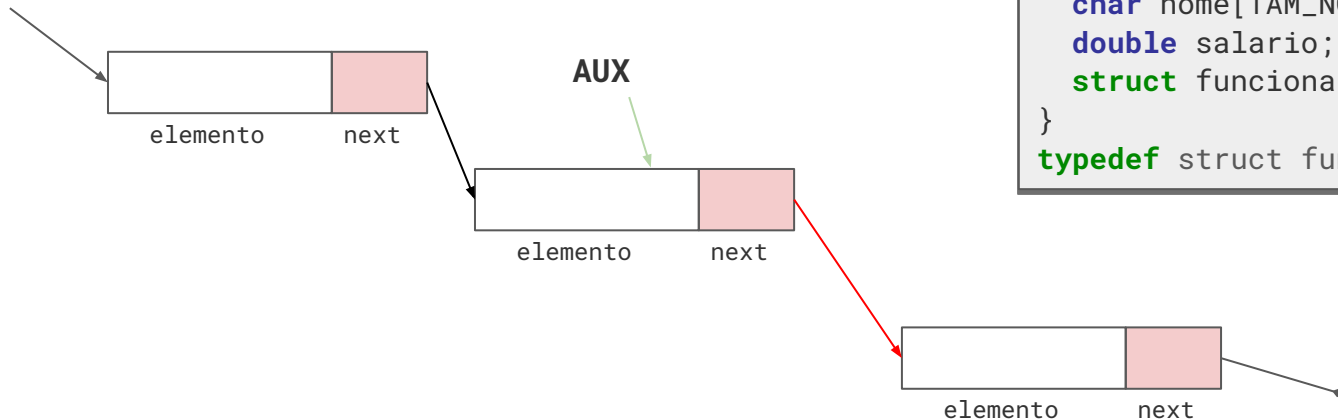
struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;

```

```
Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
}
```

Faz o aux apontar para o próximo

first



```
struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;
```

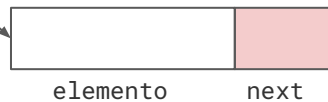
```

Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
}

```

Faz o aux apontar para o próximo

first



AUX



```

struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;

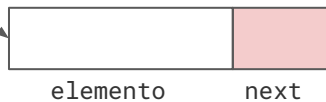
```

```
Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
}
```

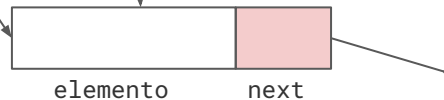
Imprime o conteúdo de AUX

```
printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
```

first



AUX



```
struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;
```

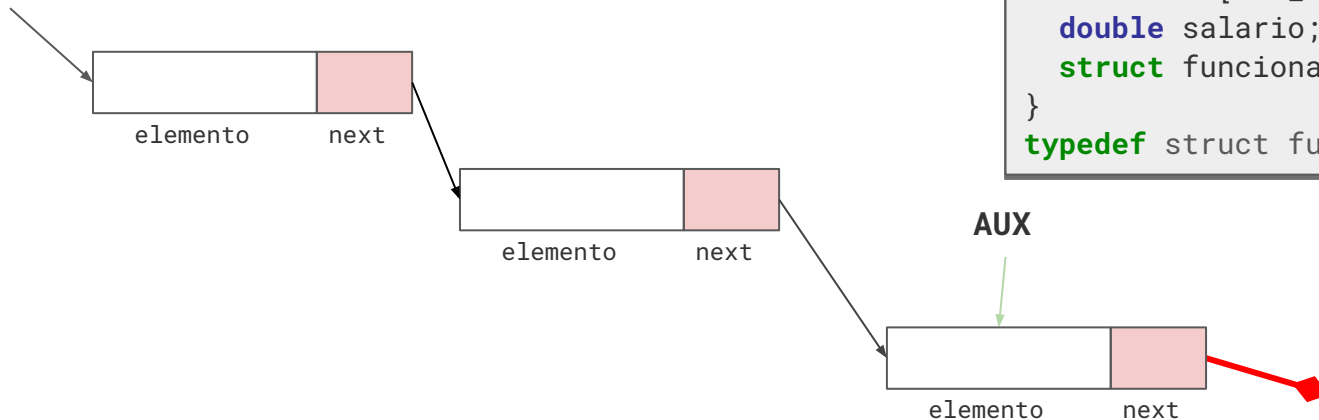
```

Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
}

```

Faz o aux apontar para o próximo

first



```

struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;

```

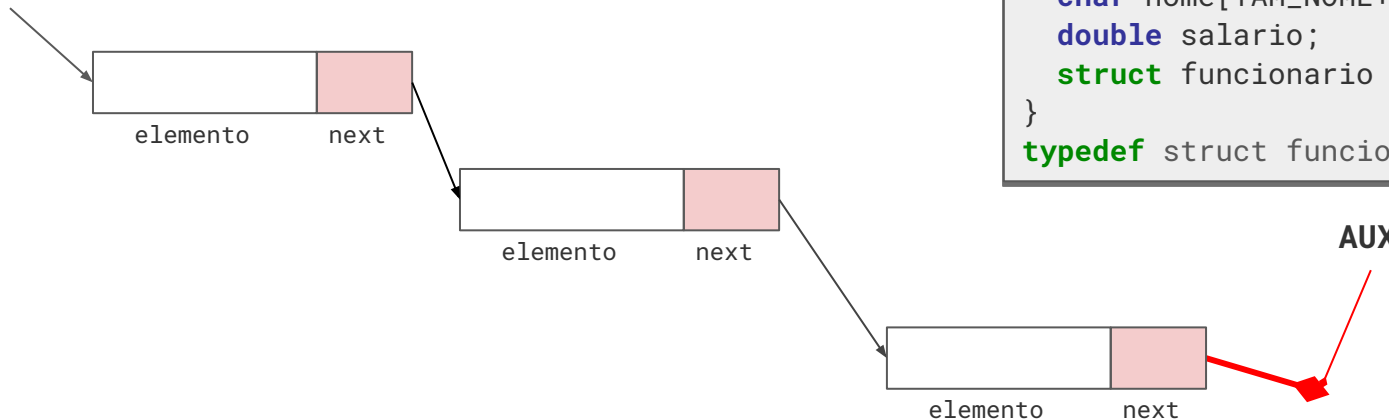
```

Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %lf\n", aux->id, aux->nome, aux->salario);
}

```

Faz o aux apontar para o próximo

first



```

struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;

```



```

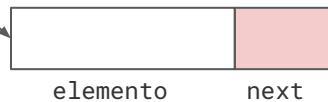
Funcionario *aux; //vai ser nosso 'contador'
for (aux = first; aux != NULL; aux = aux->next){
    //aqui aux vale o elemento atual na lista.
    printf("Funcionario id: %d, nome: %s, salario: %f\n",
aux->id, aux->nome,
}

```

Agora AUX é NULL.

aux->salario);

first



AUX

```

struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;

```

Lista encadeada simples

- Incluir um elemento na lista
 - No fim (+simples)

```
struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;
```

```
Funcionario *insEnd(Funcionario *first, Funcionario *f)
{
    Funcionario *aux;
    if (first==NULL) // lista vazia
    {
        first=f;
        return first;
    }
    for (aux=first; aux->next!=NULL; aux=aux->next); // percorre a lista até o último
    aux->next=f; // o último aponta agora para o novo
    return first;
}
```

Lista encadeada simples

- Incluir um elemento na lista (sem return)
 - No fim (+simples)
 - precisa usar ponteiro para ponteiro

```
struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;
```

```
void insEnd(Funcionario **first, Funcionario *f)
{
    Funcionario *aux;
    if (*first==NULL) // lista vazia
    {
        *first=f;
    }
    for (aux=*first;aux->next!=NULL;aux=aux->next); // percorre a lista até o último
    aux->next=f; // o último aponta agora para o novo
}
```

//Chamar a função passando o endereço do ponteiro como argumento.

```
insEnd(&first, f);
```

first e f devem ser substituídos pelos nomes das variáveis criadas na main()

Lista encadeada simples

- Incluir um elemento na lista
 - Em qualquer posição (+ complexo)

```
struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;
```

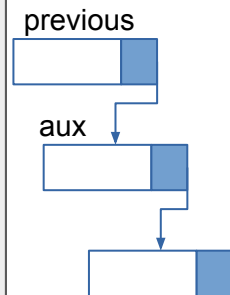
```
Funcionario *insAfter(Funcionario *first, Funcionario *f, int id)
{
    Funcionario *aux;
    if (first==NULL){ // lista vazia (início igual)
        first=f;
        return first;
    }
    for (aux=first;aux->next!=NULL;aux=aux->next) { // percorre a lista até encontrar id
        if (aux->id==id) break; // encontrou a posição para inserir depois (sai do laço)
    }
    f->next=aux->next; // o novo deve ser inserido depois do aux
    aux->next=f;       // o próximo do aux aponta para o novo
    return first;
}
```

Lista encadeada simples

- Excluir um elemento da lista
 - Encontrar endereço e refazer o aponteiamento
 - Liberar a memória do elemento excluído

```
struct funcionario{
    int id;
    char nome[TAM_NOME+1];
    double salario;
    struct funcionario *next;
}
typedef struct funcionario Funcionario;
```

```
Funcionario *delFunc(Funcionario *first, int id)
{
    Funcionario *aux, *previous;
    for (aux=first; aux!=NULL; aux=aux->next) {
        if (aux->id==id) // elemento encontrado
        {
            if (aux==first) {
                first=first->next; // só atualiza o first
                break;
            }
            previous->next=aux->next; // faz o anterior do aux apontar para o próximo dele
            break;
        }
        previous=aux; // aponta para a endereço antes do aux
    }
    if (aux!=NULL) free(aux); // CUIDADO: pode ser que o funcionário não existe
    return first;
}
```



Exercício

- A função `Funcionario *insAfter(Funcionario *first, Funcionario *f, int id)` inclui um novo elemento na lista após a posição do elemento com o valor do id passado como parâmetro (slide 20)
- Você deverá implementar a função `Funcionario *insBefore(Funcionario *first, Funcionario *f, int id)` que irá inserir o elemento na posição anterior (antes) da posição do elemento com o valor do id.