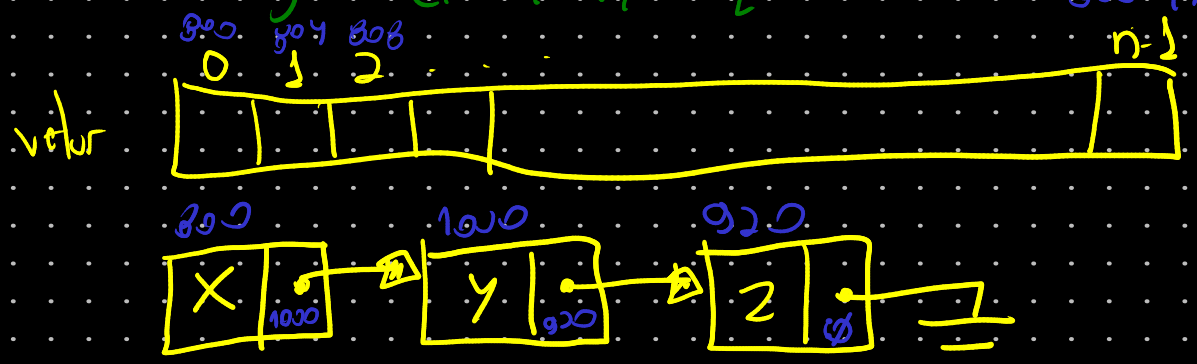
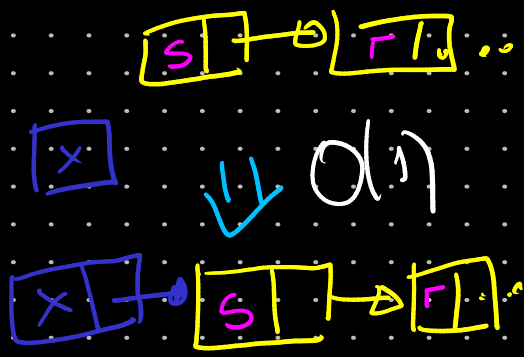
Lista encadeada

- É uma lista linear que não armazenada de forma contígua em memória

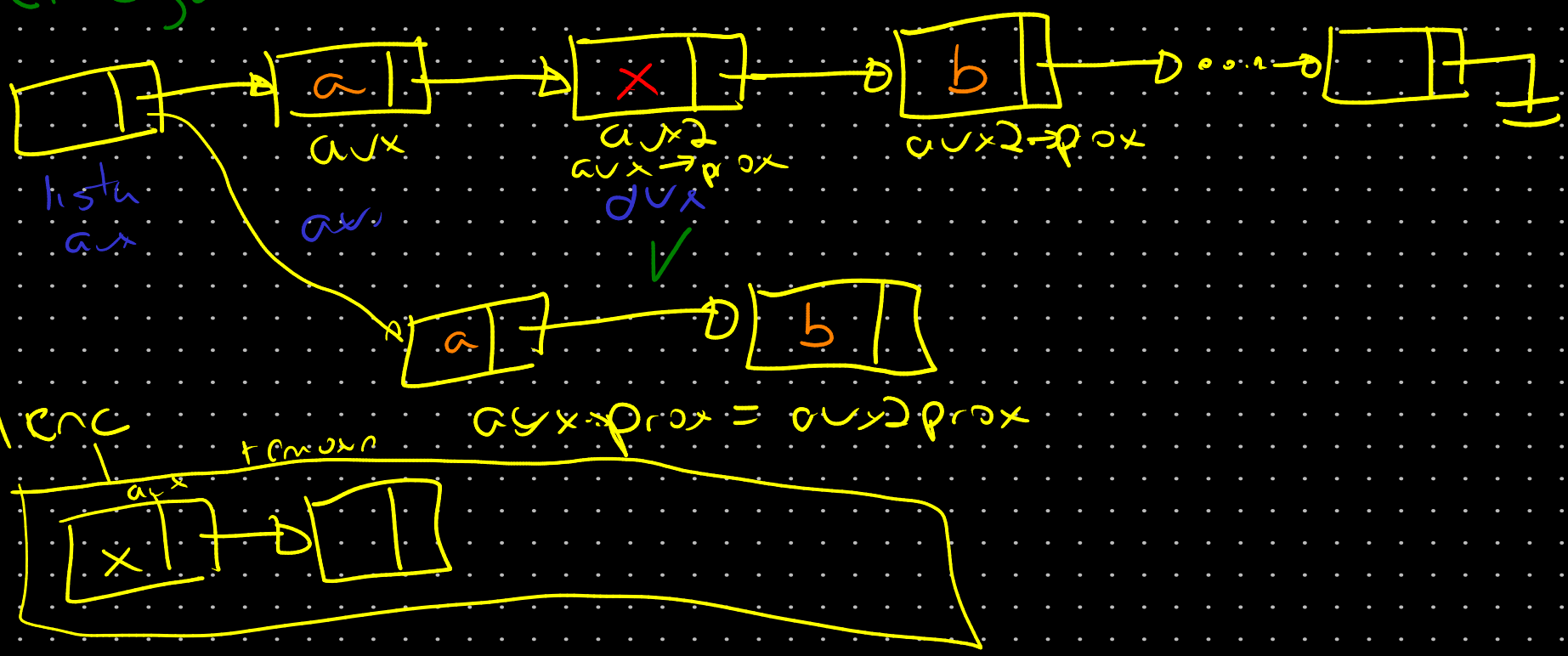


→ contíguo

Chamamos uma lista linear não contígua de lista encadeada porque cada elemento "aponta" para o próximo formando uma "cadeia": LINKED LIST

Operação	Inserção	Remoção	Busca
		O(n)	O(n)
	O(1)	O(1)	

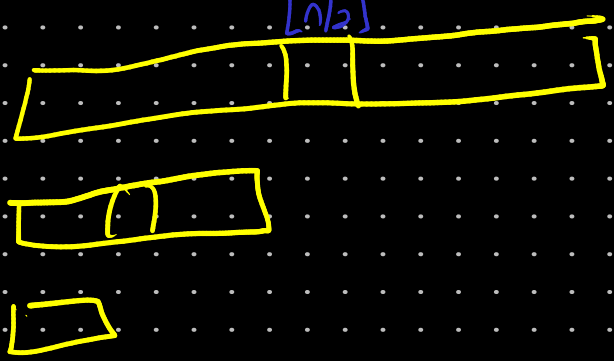
Remoção

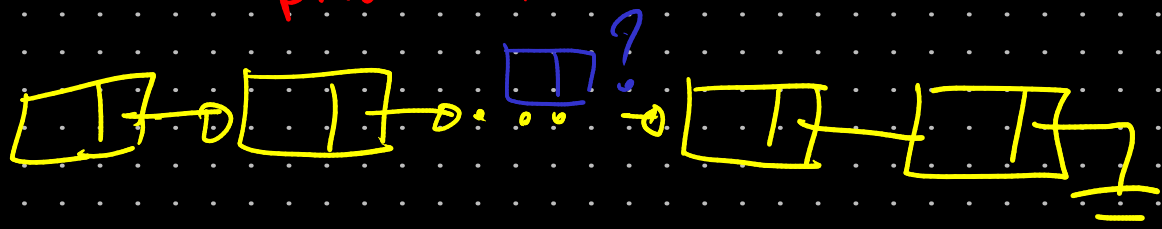

a
x
b
aux
aux2
aux->prox
dux
lista
aux
aux
a
b
aux->prox = aux2->prox
lenc
remover
aux
x

Vantagens e Desvantagens

Vantagens ✓
- pode ser usada em memória fragmentada
- não precisa alocar previamente toda a memória
- economiza memória (só está em memória dados usados)

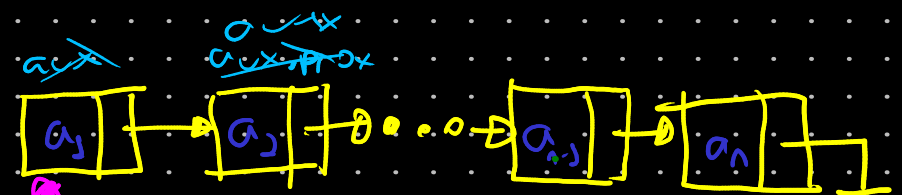
Desvantagens ✗
- custo extra para o encadeamento
- busca é sempre linear (busca binária não é aplicável)





Implementação

nó { Estrutura
valor int
prox int
}



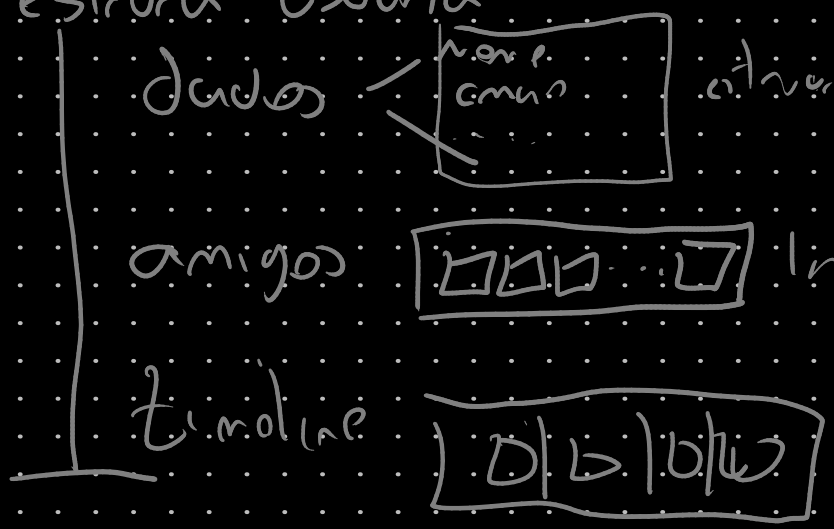
lista "cabeça"

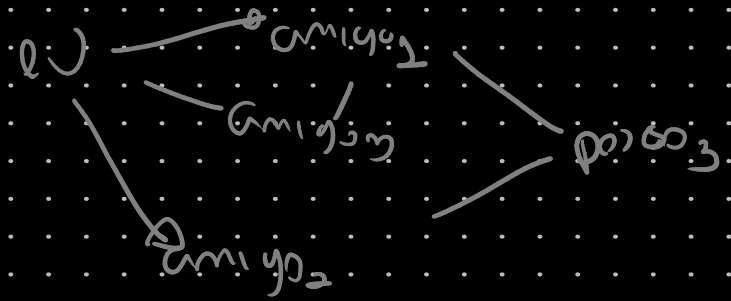
λ → elemento nulo

```
função busca (x)
    aux ← lista
    enquanto aux ≠ λ faça
        se aux.valor = x então
            retorna aux
        senão
            aux = aux.prox
        fimse
    fimenquanto
    "elemento não encontrado"
```

estrutura fixa complexa

FACE BOOK

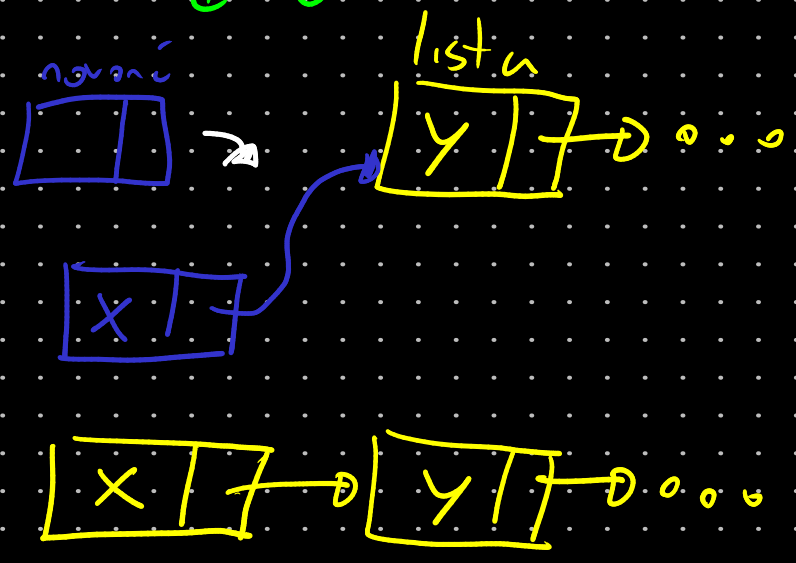




Inserção (Simples)

Para inserir um elemento, inserimos sempre no início

```
função insere(x)
    cria(novo_nó)
    novo_nó.valor ← x
    novo_nó.prox ← lista
    lista ← novo_nó
```

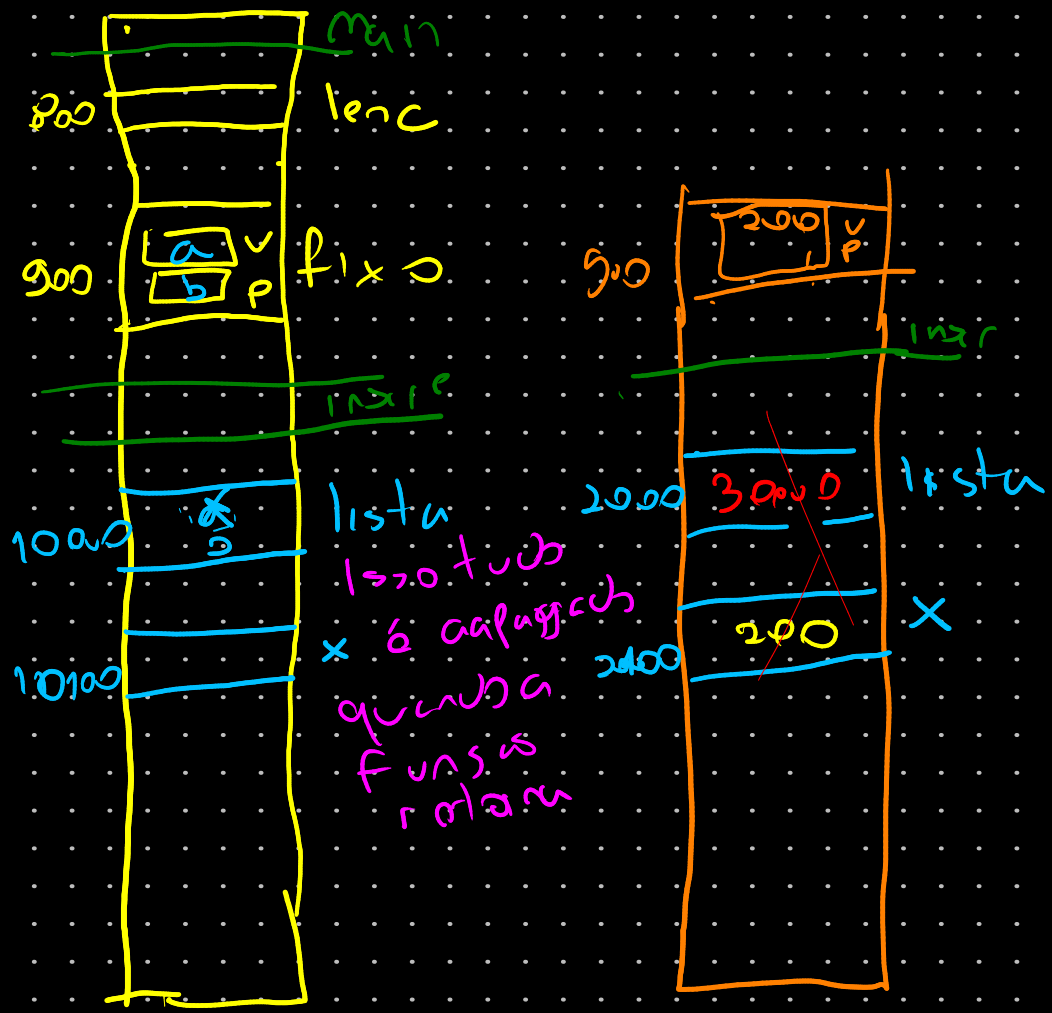


Parênteses → Porque usamos ponteiro para ponteiro na implementação em C

```
struct nodo {
    int valor;
    struct nodo *prox;
};
```

```
struct nodo *lenc; //lenc == NULL
struct nodo fixo
```

```
int insere(struct nodo *lista,
           int x) { ... }
```




```
insere(fixo, 200);
```

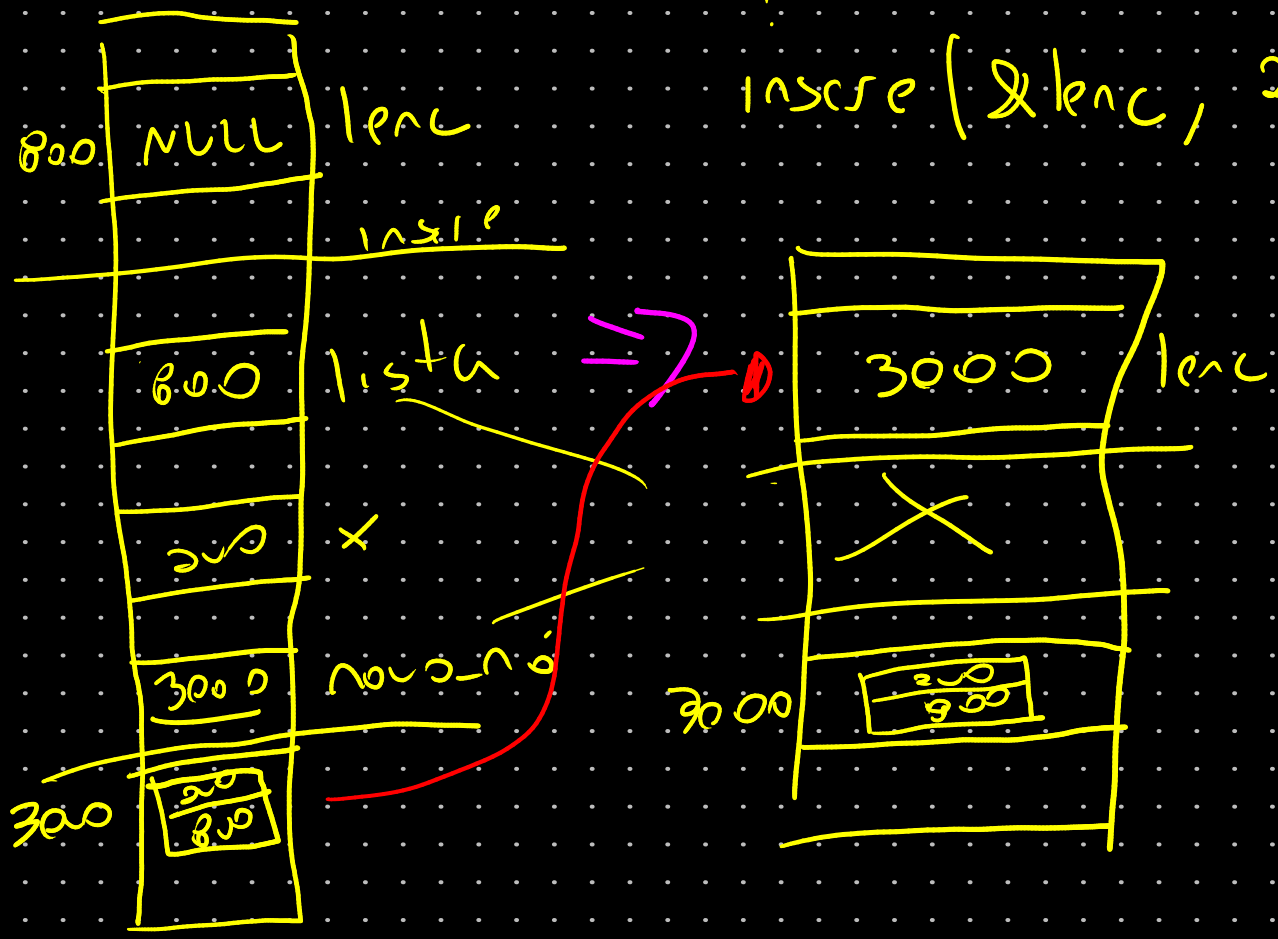
```
int insere (struct nodo *lista,
            int x)
```

```
insere (&fixo, 200)
```

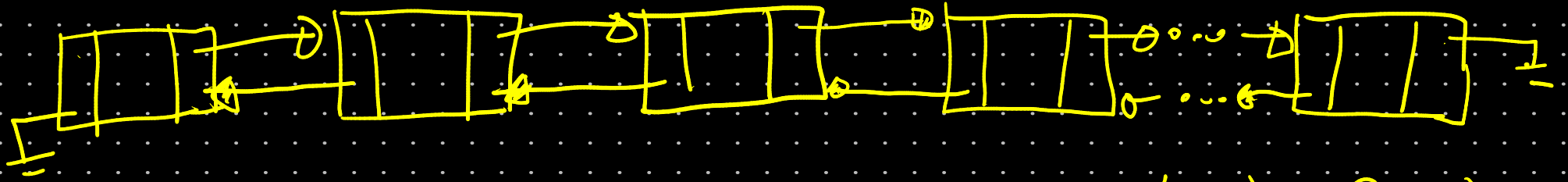
Cópia

insere(struct noeud **liste, int x){ ... }

insere(&lenc, 200)



Lista Duplamente Encadeada



Lista Encadeada Circular

Lista Duplamente Encadeada Circular