UNIVERSIDADE FEDERAL DE ITAJUBÁ

Algoritmos e Estrutura de Dados I

CTCO-01

Estruturas Encadeadas
Pilha

Vanessa Cristina Oliveira de Souza

Tipo Abstrato de Dados do tipo PILHA

Pilha

- Estrutura de Dados
  - LIFO
  - O último elemento a ser inserido será o primeiro elemento a ser retirado
- Operações
  - CriaPilha (cria a estrutura pilha vazia)
  - Push (inserir elemento na pilha)
  - Pop (remover elemento da pilha)
  - Top (mostrar quem está no topo da pilha)
  - Esvazia (remove todos os elementos da pilha)
  - pilhaVazia (verifica se a pilha está vazia)
  - pilhaCheia (verifica se a pilha está cheia – estruturas estáticas)
- Implementação
  - Vetor
  - Lista

Pilha

- Qualquer estrutura desse tipo possui um ponteiro denominado TOPO, na qual todas as operações de inserção e remoção acontecem.

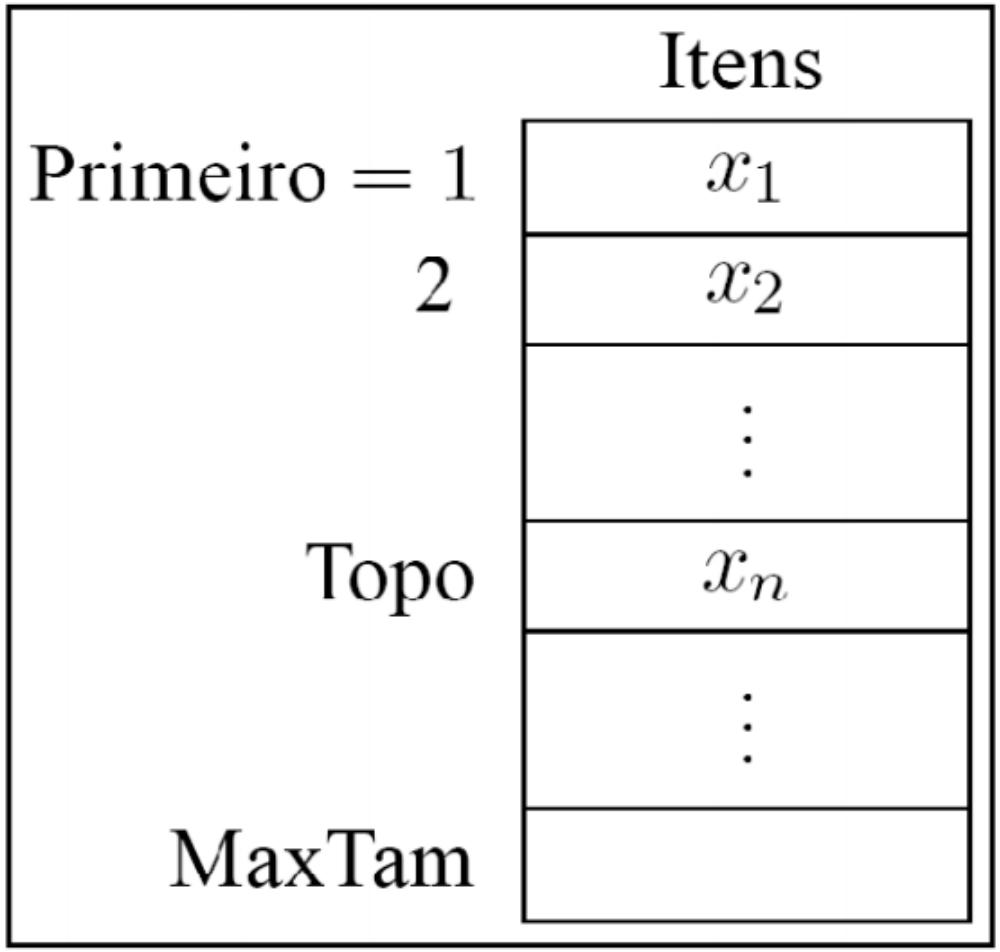
- As operações acontecem sempre na mesma extremidade da estrutura.

PILHA ESTÁTICA

Pilhas Estáticas

- Em uma implementação por meio de arranjos (vetores) os itens da pilha são armazenados em posições contíguas de memória.
  - Os itens são armazenados em um arranjo de tamanho suficiente para conter a pilha
  - Há uma referência para o item no topo da pilha
  - Há uma constante para guardar o tamanho máximo permitido na pilha

Pilhas Estáticas



- **Vetor (alocação sequencial)**
- **Topo**
- **MaxTam**

PILHA DINÂMICA

Estruturas Sequenciais

- Nas estruturas lineares sequenciais, os elementos são alocados de uma só vez e de forma sequencial na memória.

	0	1	2	3	4	5	6
V	15	6	2	9	8	3	0

A posição 3 do vetor V é acessada diretamente porque seu endereço na memória é conhecido

V[3] = V[0] + sizeof(int)

Isso só é possível porque os elementos estão alocados na memória de forma contígua, ou sequencial

Estruturas Sequenciais

- Nas estruturas lineares sequenciais, os elementos são alocados de uma só vez e de forma sequencial na memória.

	0	1	2	3	4	5	6
V	15	6	2	9	8	3	0

A posição 3 do vetor V é acessada diretamente porque seu endereço na memória é conhecido

V[3] = V[0] + sizeof(int)

Esse comportamento é independente da forma de alocação : estática ou dinâmica

Estruturas Encadeadas

- Diferentemente das estruturas lineares sequenciais (ou contíguas), os elementos **não estão necessariamente armazenados sequencialmente na memória.**

- Ou seja, os elementos são alocados em momentos diferentes da execução do programa e, consequentemente, em endereços de memória diferentes.

- Neste caso, **não é possível** acessar um elemento da forma V[3] = V[0] + sizeof(int)

Estruturas Encadeadas

- Sendo assim, como é possível acessar os demais elementos da estrutura???

Estruturas Encadeadas

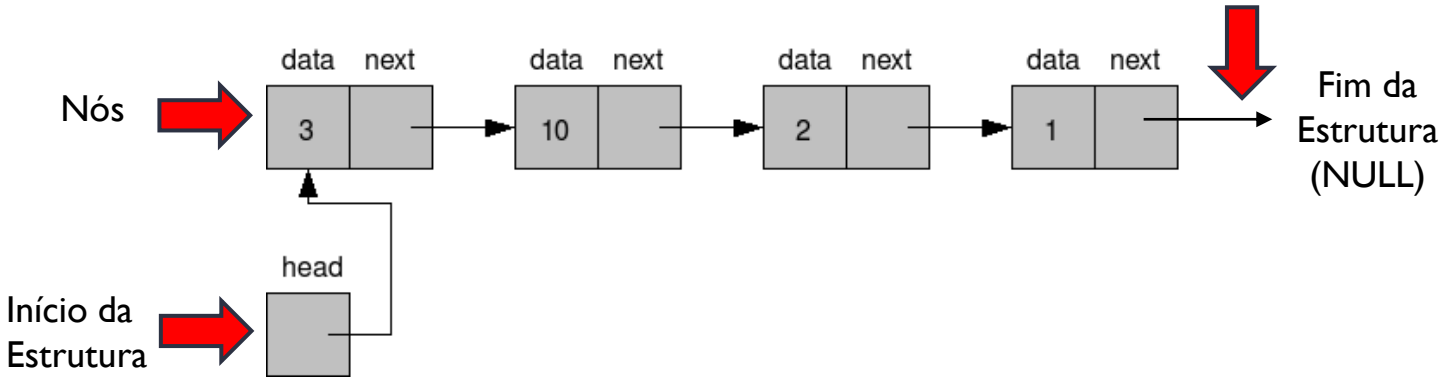
- Sendo assim, como é possível acessar os demais elementos da estrutura???

- Para manter a ordem lógica entre os elementos, é necessário além do espaço para armazenamento da informação, um espaço para armazenar uma **referência da localização na memória** onde o próximo elemento da estrutura (ou o anterior) se encontra.

Estruturas Encadeadas

- TODA ESTRUTURA ENCADEADA É FORMADA POR NÓS

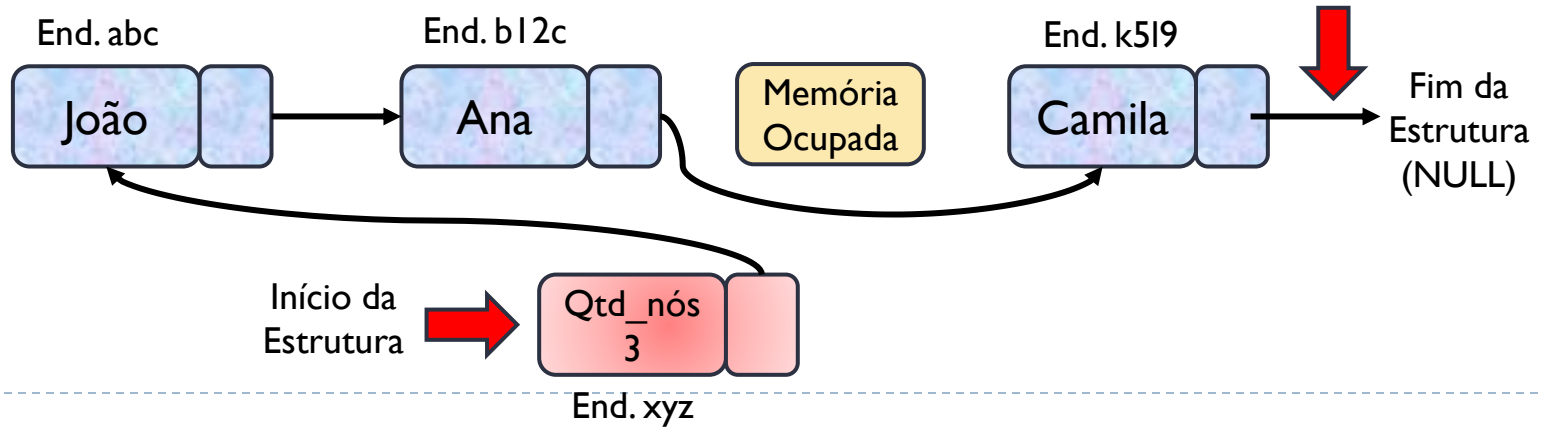
- TODO NÓ É FORMADO PELAS VARIÁVEIS QUE ARMAZENA (DADOS) E UM PONTEIRO QUE É RESPONSÁVEL PELO ENCADEAMENTO.



Estruturas Encadeadas

- TODA ESTRUTURA ENCADEADA É FORMADA POR NÓS

- TODO NÓ É FORMADO PELAS VARIÁVEIS QUE ARMAZENA (DADOS) E UM PONTEIRO QUE É RESPONSÁVEL PELO ENCADEAMENTO.



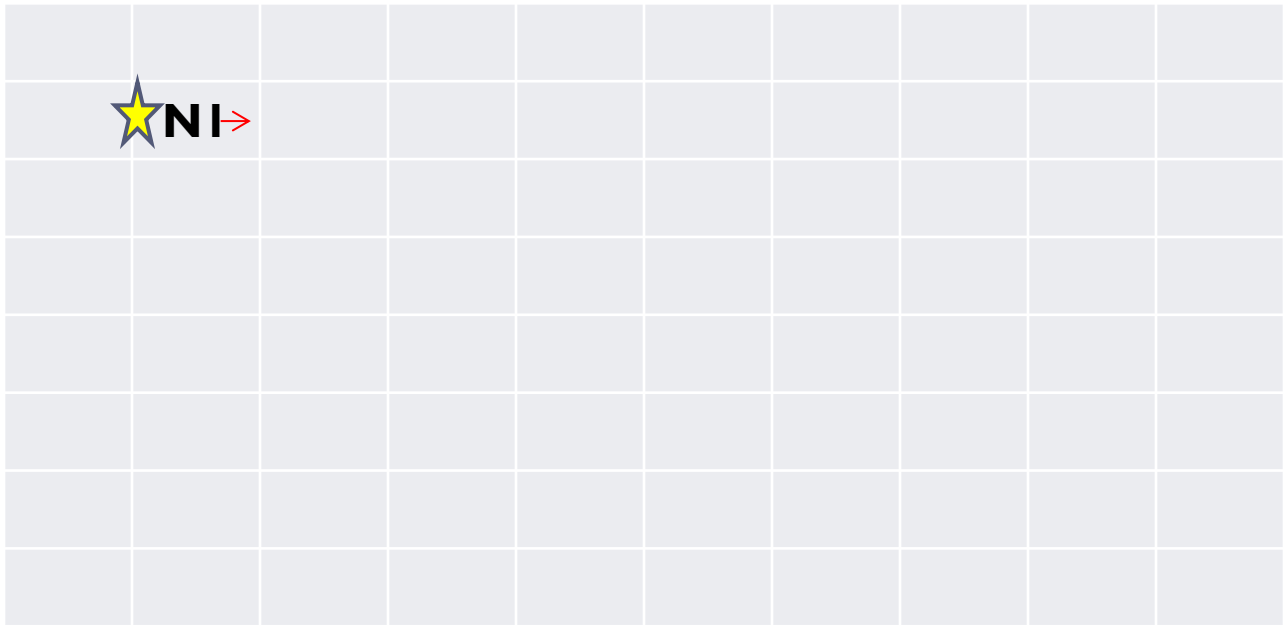
Estruturas Encadeadas

- Os elementos podem ocupar quaisquer células de memória (não necessariamente consecutivas) e, para manter a relação de ordem linear, juntamente com cada elemento é armazenado o endereço do próximo elemento da lista.

- Os elementos são armazenados em blocos de memória denominados **nós**, sendo que cada nó é composto por dois campos:
  - um para armazenar dados e
  - outro para armazenar endereço.

nó

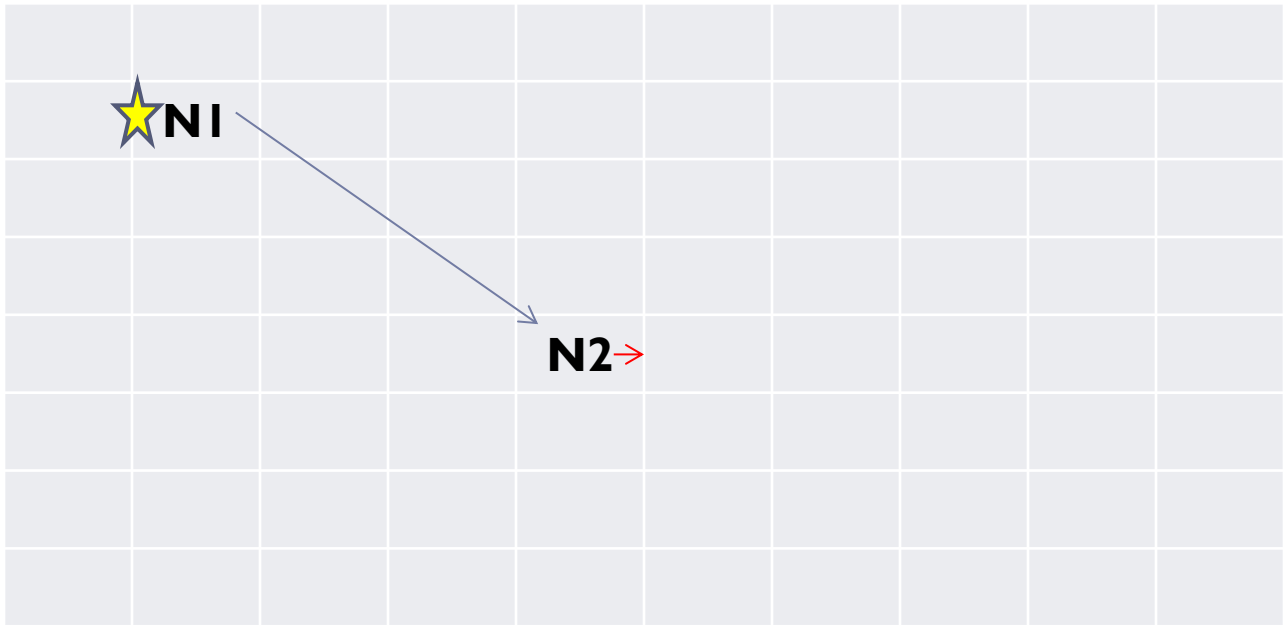
Estruturas Encadeadas



→ NULL

★ Início da Estrutura

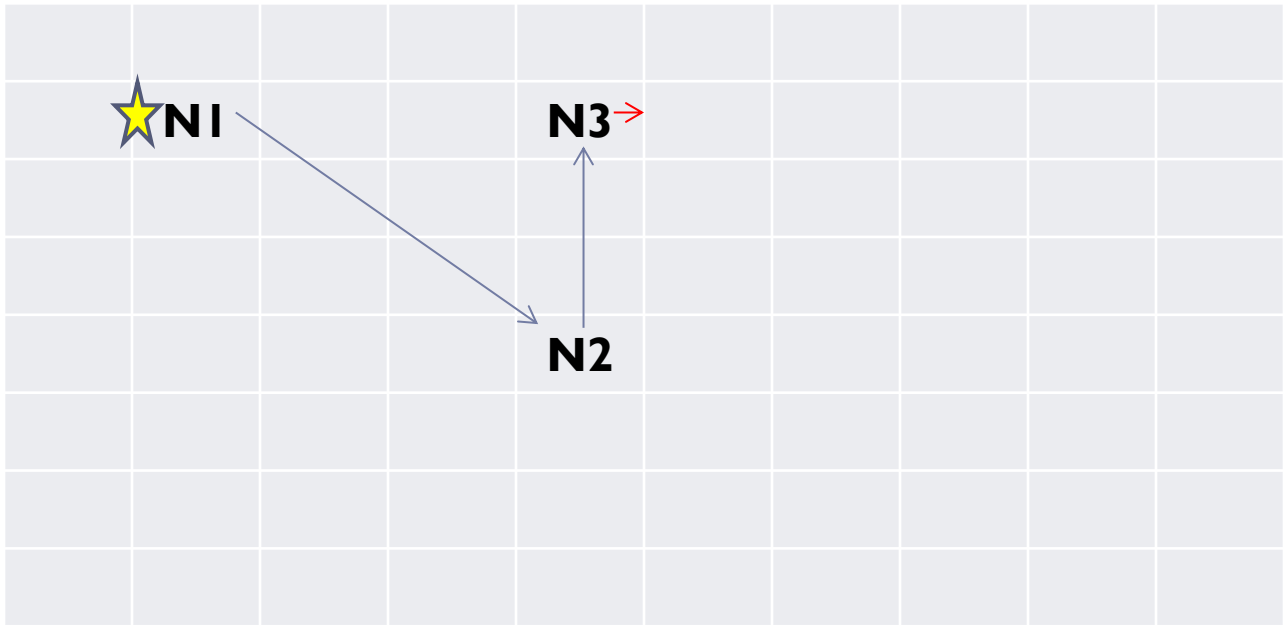
Estruturas Encadeadas



NULL

Início da Estrutura

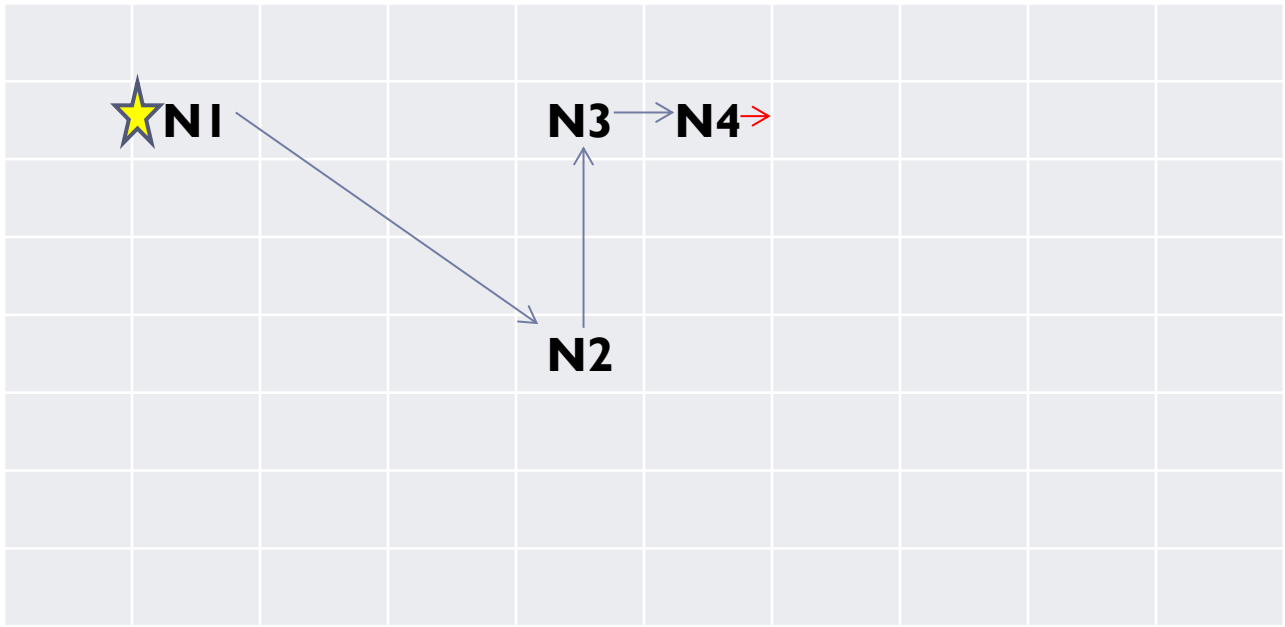
Estruturas Encadeadas



→ NULL

★ Início da Estrutura

Estruturas Encadeadas



→ NULL

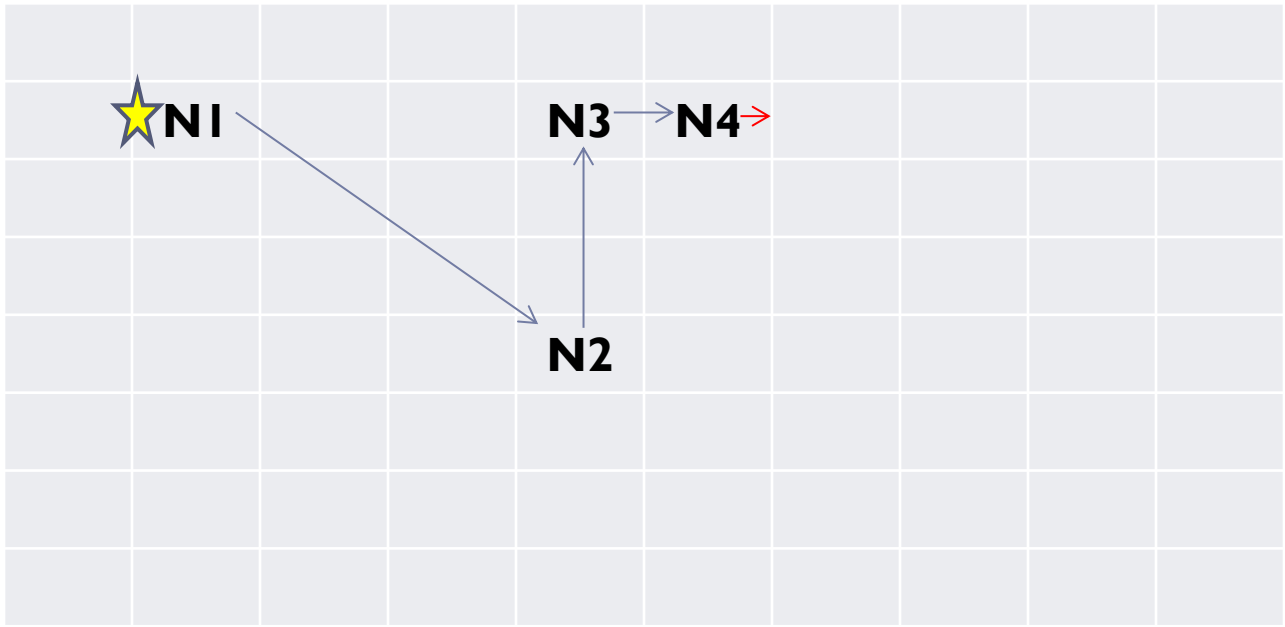
★ Início da Estrutura

Estruturas Encadeadas

- **A ordem lógica entre os elementos é toda feita por meio dos ponteiros.**

- Exemplo :
  - Inserir um elemento (nó) entre os nós N1 e N2

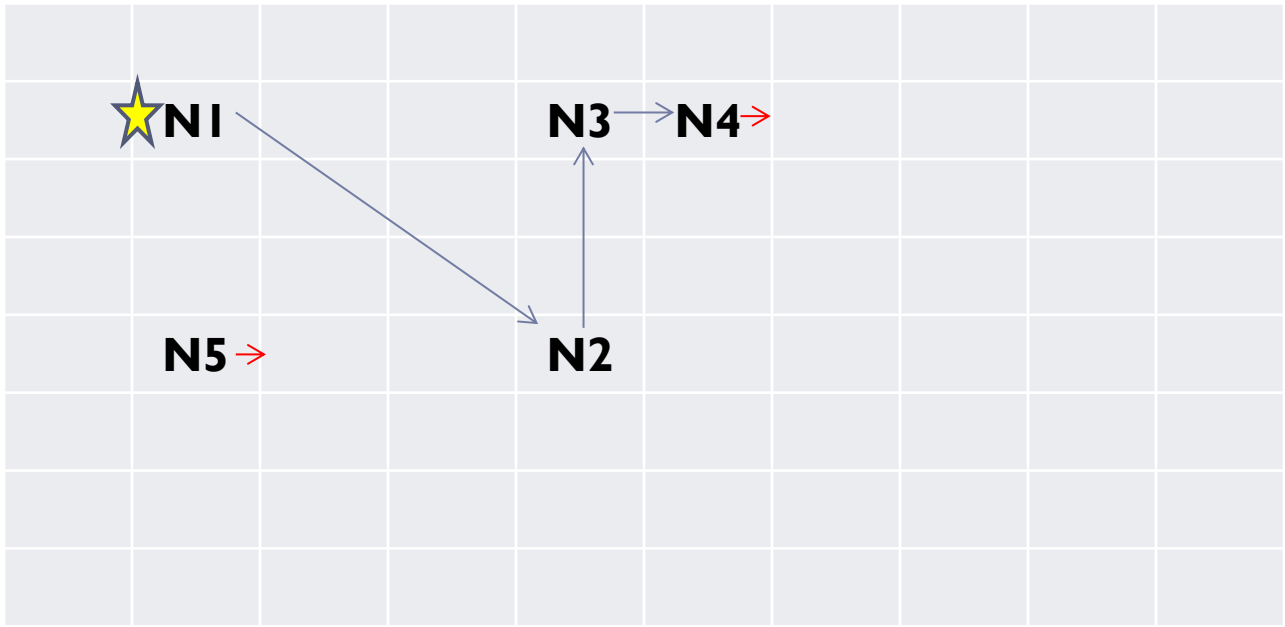
Estruturas Encadeadas



→ NULL

★ Início da Estrutura

Estruturas Encadeadas

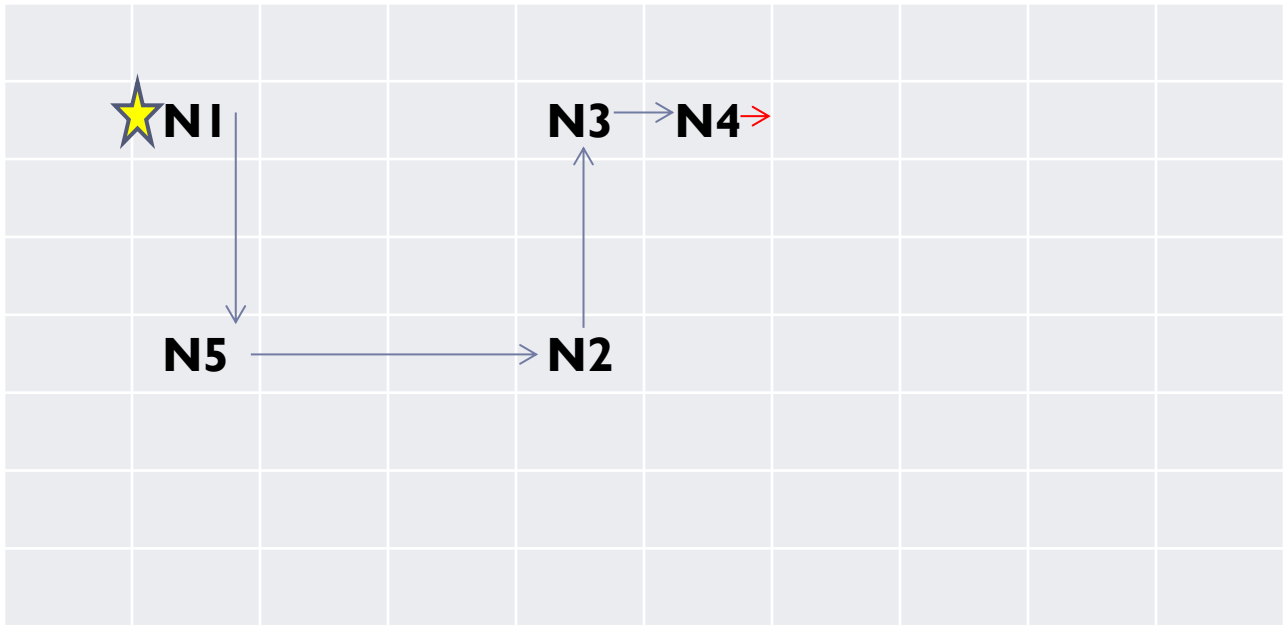


CRIA O NÓ

→ NULL

★ Início da Estrutura

Estruturas Encadeadas



INSERE NA ESTRUTURA AJUSTANDO OS PONTEIROS

→ NULL

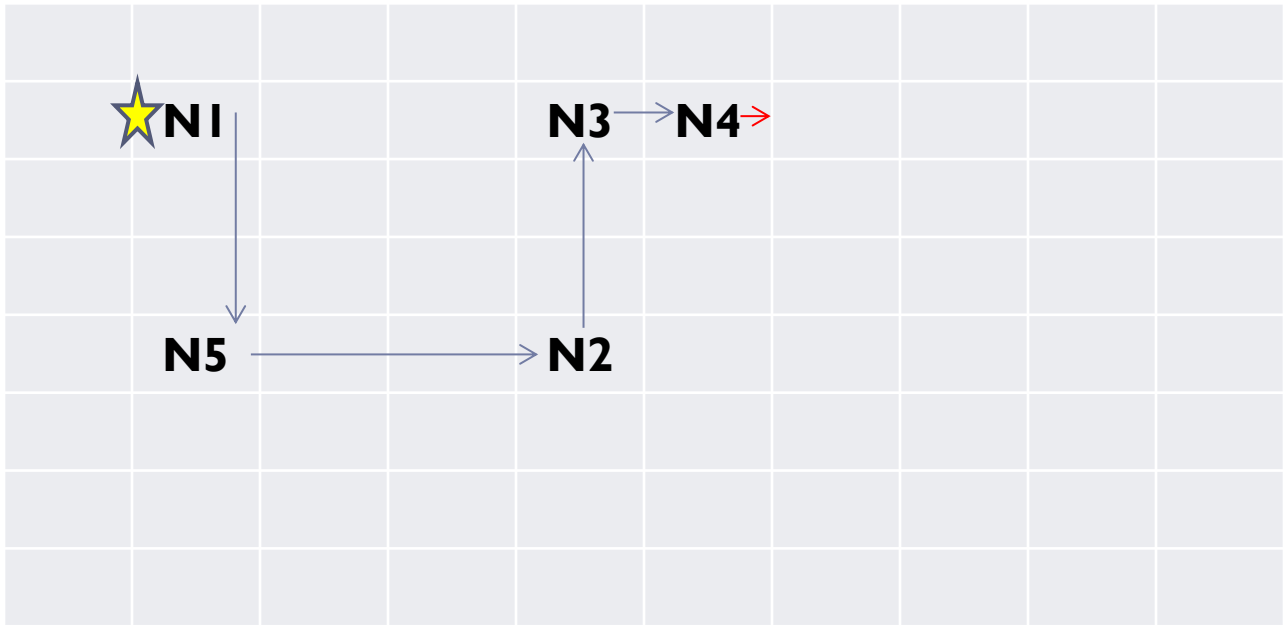
★ Início da Estrutura

Estruturas Encadeadas

- A ordem lógica entre os elementos é toda feita por meio dos ponteiros.

- Exemplo :
  - Remover o elemento N3

Estruturas Encadeadas

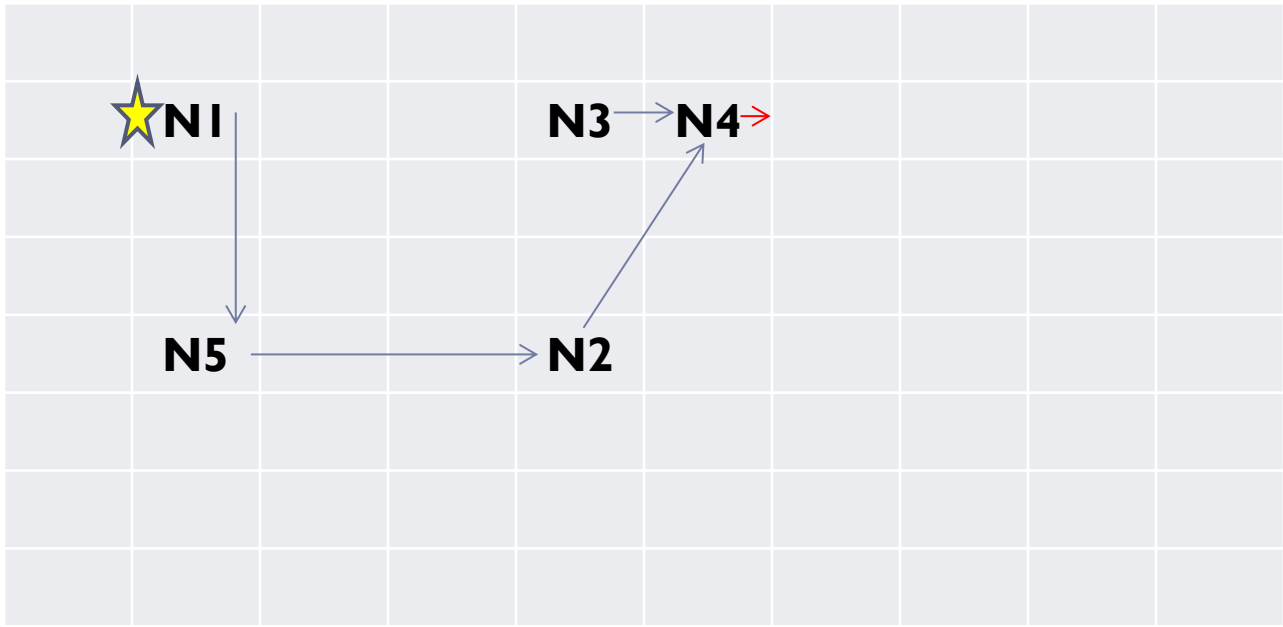


REMOVE O NÓ DA ESTRUTURA AJUSTANDO OS PONTEIROS

→ NULL

★ Início da Estrutura

Estruturas Encadeadas

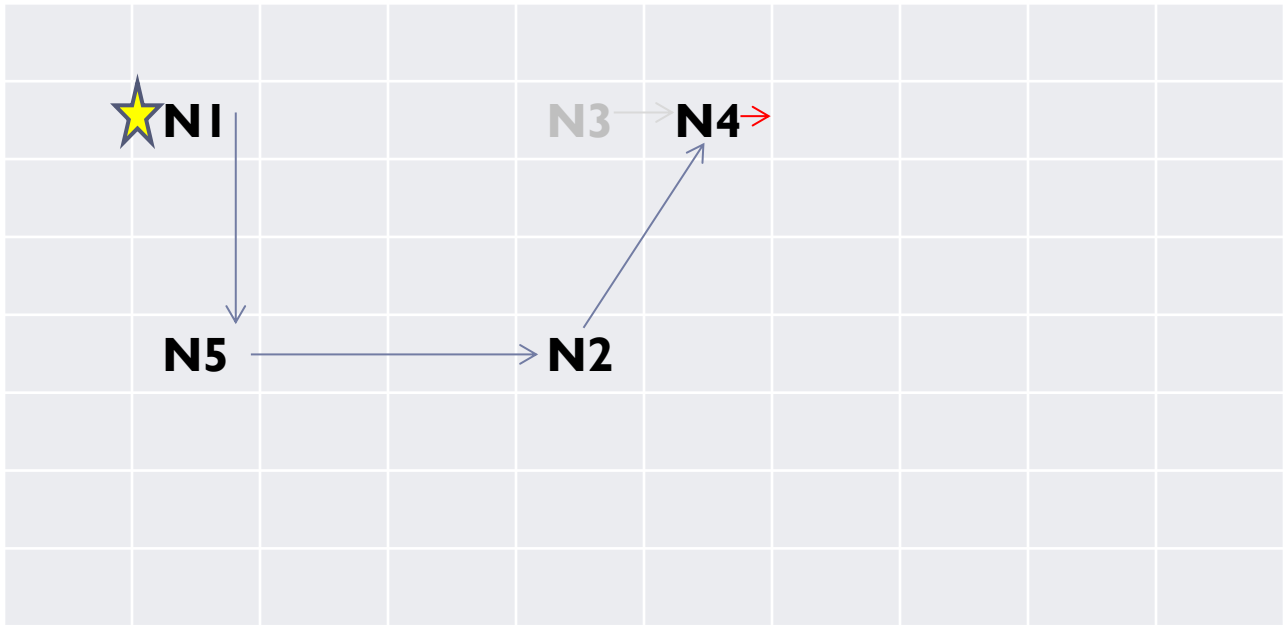


REMOVE O NÓ DA ESTRUTURA AJUSTANDO OS PONTEIROS

→ NULL

★ Início da Estrutura

Estruturas Encadeadas



LIBERA A MEMÓRIA DE N3

→ NULL

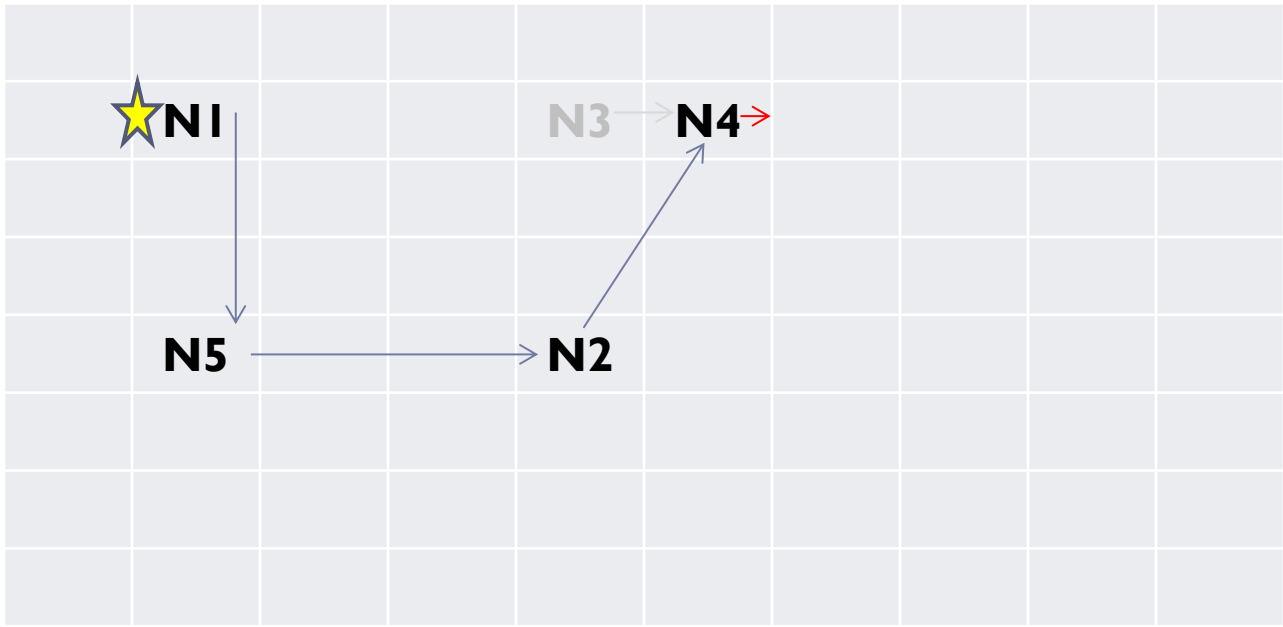
★ Início da Estrutura

Estruturas Encadeadas

- A ordem lógica entre os elementos é toda feita por meio dos ponteiros.

- Exemplo :
  - Remover o elemento N1

Estruturas Encadeadas

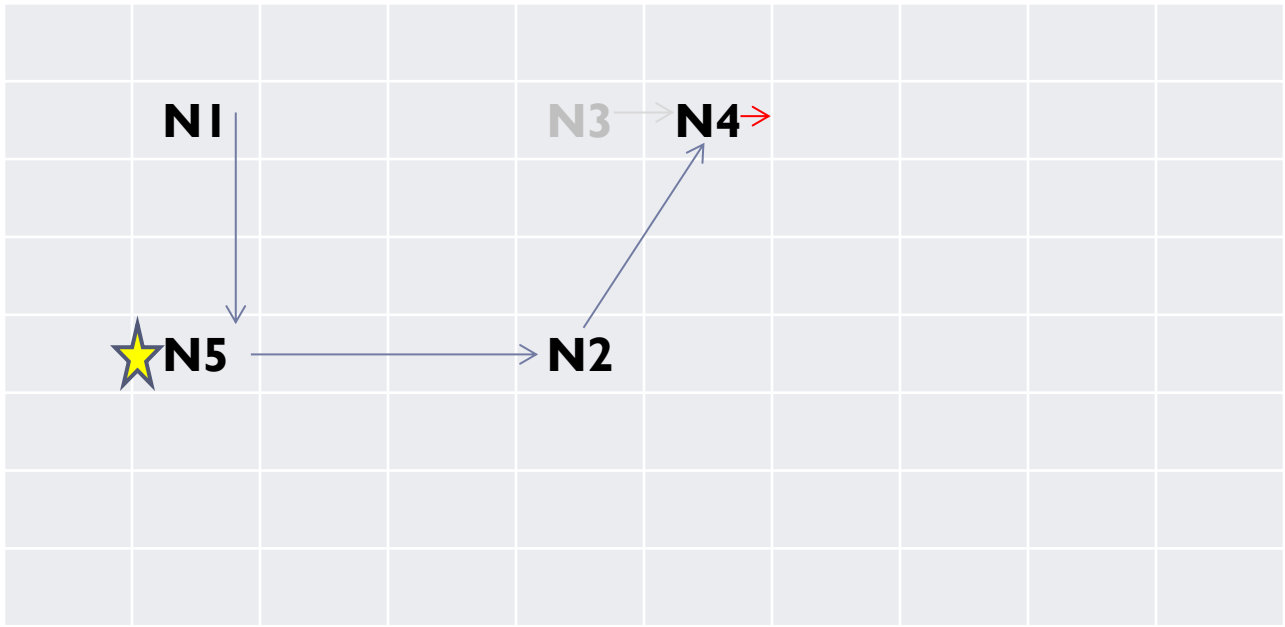


N1 marca o início da estrutura

→ NULL

★ Início da Estrutura

Estruturas Encadeadas

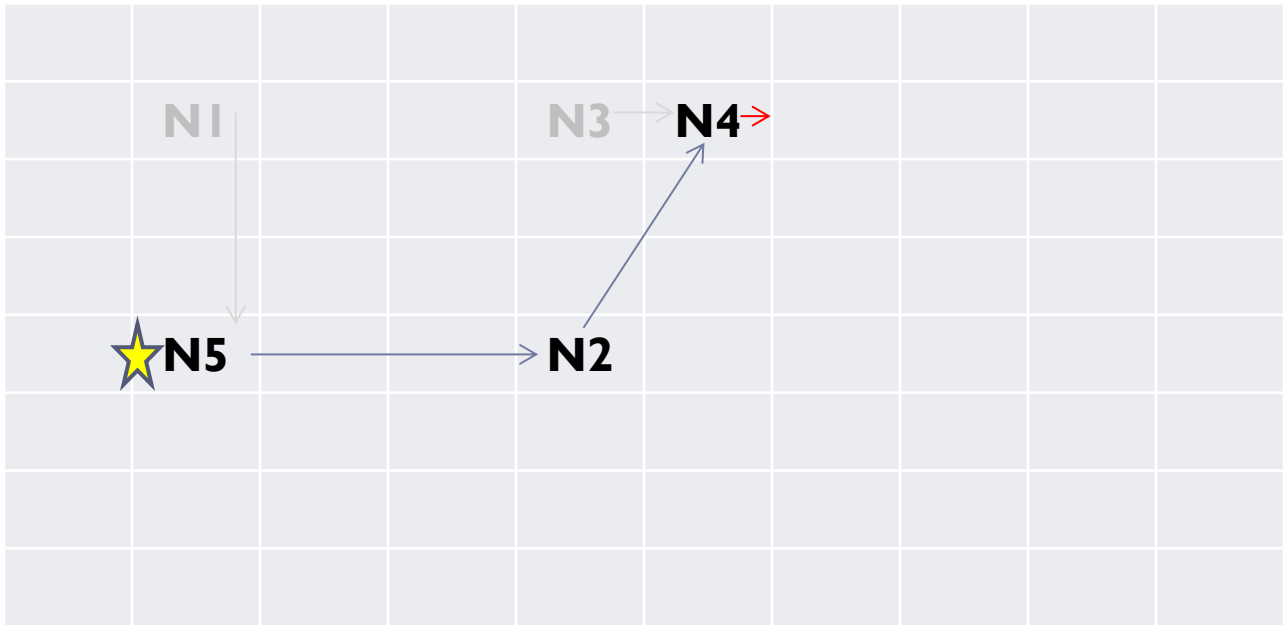


AJUSTA A LÓGICA

→ NULL

★ Início da Estrutura

Estruturas Encadeadas

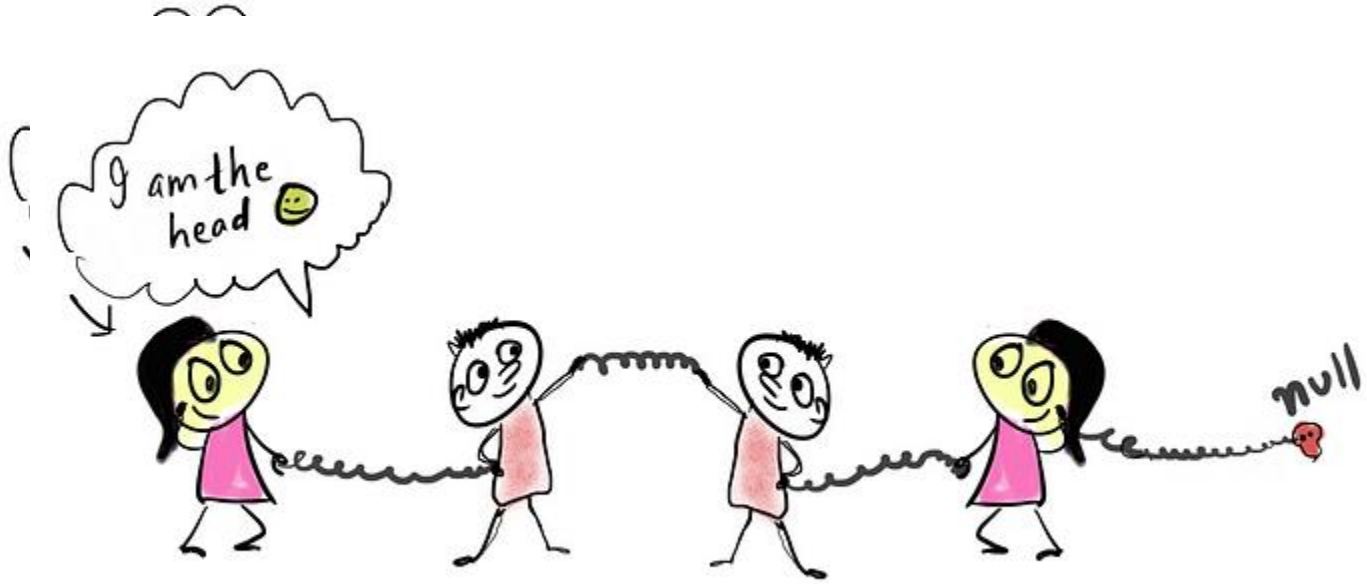


LIBERA A MEMÓRIA DE N1

→ NULL

★ Início da Estrutura

Hora de brincar



Implementação Estrutura e Nó

Estruturas Encadeadas

- TODA ESTRUTURA ENCADEADA É FORMADA POR NÓS
  - STRUCT da estrutura
    - Pilha
    - Fila
    - Lista
    - Árvore

- TODO NÓ É FORMADO PELAS VARIÁVEIS QUE ARMAZENA (DADOS) E UM PONTEIRO QUE É RESPONSÁVEL PELO ENCADEAMENTO.
  - STRUCT nó

Estruturas Encadeadas

- Os elementos são armazenados em blocos de memória denominados **nós**, sendo que cada nó é composto por dois campos:
  - um para armazenar dados e
  - outro para armazenar endereço.

nó

- Um nó tem, portanto, um **ponteiro** para o **próximo nó** (ou o anterior).

- Quando não existe o próximo nó, esse ponteiro é **inicializado** com o valor nulo
  - Um ponteiro pode ter o valor especial NULL
  - A macro NULL está definida na interface stdlib.h e seu valor é 0 na maioria dos computadores.

Estruturas Encadeadas

- Um nó tem, portanto, um **ponteiro** para o **próximo nó** (ou o anterior).

nó

```
struct no
{
    int valor;
    struct no *prox;
};
```

Estruturas Encadeadas

- Um nó tem, portanto, um **ponteiro** para o **próximo nó** (ou o anterior).

nó

```
struct pessoa
{
    char nome[30];
    int idade;
    char endereco[30];
}pessoa;

struct no
{
    struct pessoa pessoa;
    struct no *prox;
};
```

Estruturas Encadeadas

- A estrutura tem sempre um ponteiro que aponta para seu início.
- Ela também pode guardar outras informações, como a quantidade de elementos na estrutura.

```
struct pilha{
    int totalElementos;
    struct no *topo
}
```

Pilha Encadeada

Pilha Encadeada

- Quando o número máximo de elementos armazenados na pilha não é conhecido, devemos implementar a pilha usando uma estrutura de dados dinâmica e encadeada.

- A pilha pode ser representada simplesmente por um ponteiro para o primeiro nó da estrutura encadeada.

Pilha Encadeada

- Considerando o TAD Pilha Encadeada, vamos implementar as operações
  - CriaPilha (cria a estrutura pilha vazia)
  - Push (inserir elemento na pilha)
  - Pop (remover elemento da pilha)
  - Topo (mostrar quem está no topo da pilha)
  - Esvazia (remove todos os elementos da pilha)
  - pilhaVazia (verifica se a pilha está vazia)
  - ~~pilhaCheia (verifica se a pilha está cheia – estruturas estáticas)~~