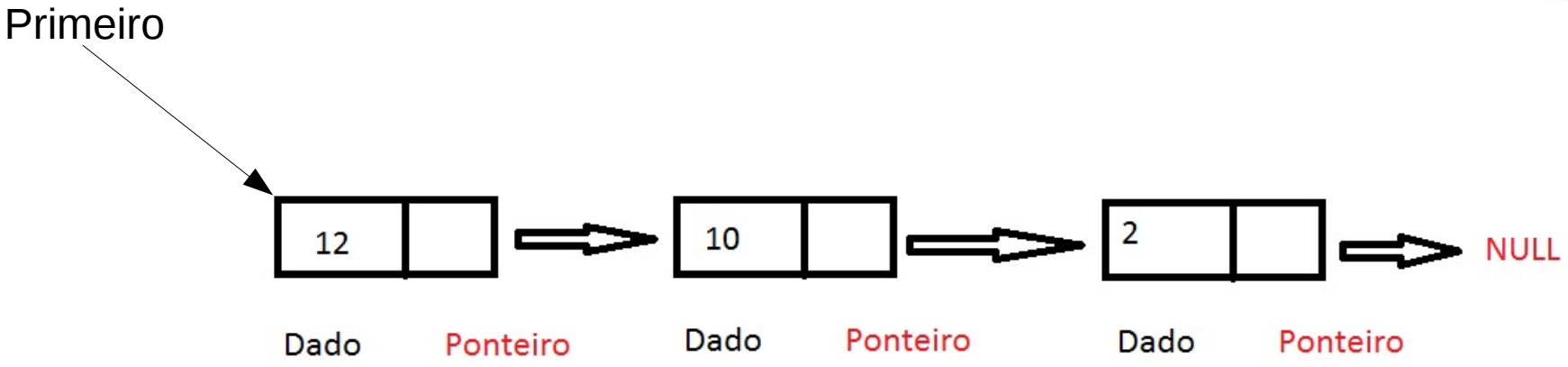
Lista simplesmente encadeada

Prof. Daniel Di Domenico
ddomenico@inf.ufsm.br

Introdução

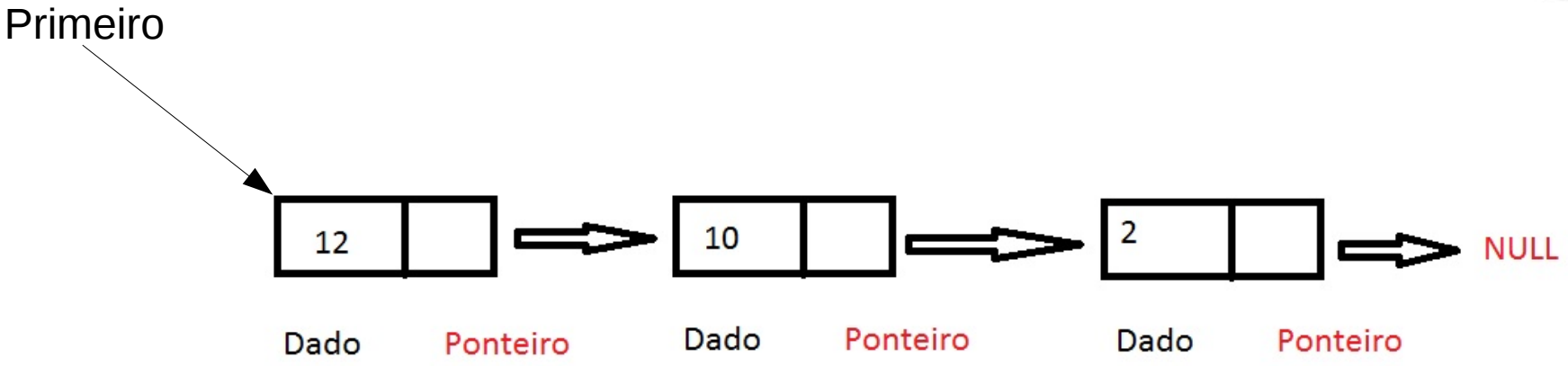
- Numa lista encadeada para cada novo elemento inserido na estrutura, alocamos um espaço de memória para armazená-lo;
- Não há como garantir que os elementos armazenados na lista ocuparão um espaço de memória contígua (vizinho), portanto não temos acesso direto aos elementos da lista:
  - Diferente de vetor;
- Para percorrer todos os elementos da lista, devemos guardar seu encadeamento, ou seja, um **ponteiro** para o próximo elemento da lista.

Exemplo



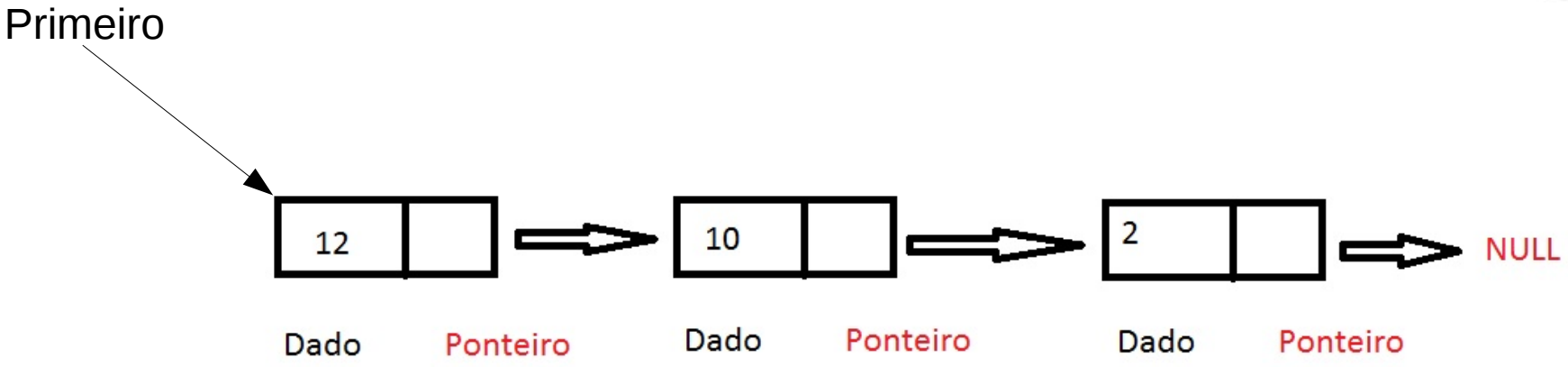
- A estrutura consiste em uma sequência encadeada de elementos, em geral chamados de nós da lista;
- Um nó da lista é representado por uma estrutura que tem dois campos:
  - 1- A informação armazenada;
  - 2- O ponteiro para o próximo elemento da lista;
- O último elemento da lista armazena, como próximo elemento, um ponteiro inválido, com valor NULL, sinalizando assim que **não** existe um próximo elemento.

Estrutura



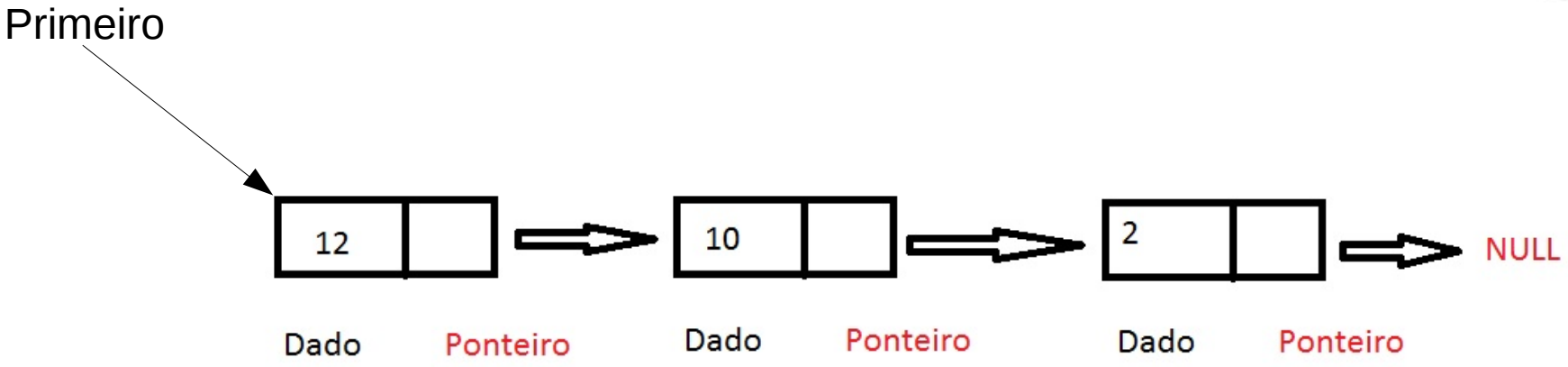

```
/* define a struct TAD pilha */
struct pilha {
    float info;              /* dado */
    struct lista *prox;  /* ponteiro para o proximo elemento */
};
```

Inserção



- Pode ser feita em qualquer posição da lista:
  - **Início:** o ponteiro do novo nó deve apontar para o primeiro elemento da lista;
  - **Outra posição:**
    - O ponteiro do novo nó deve apontar para o próximo nó (ponteiro “proximo” do nó anterior);
    - O ponteiro do nó anterior deve apontar para o novo nó;

Remoção



- Deve-se buscar o elemento que deseja-se excluir;
- Pode ser feita de qualquer posição da lista:
  - **Início:** deve-se transformar o segundo elemento no primeiro da lista;
  - **Outra posição:** o ponteiro do nó anterior deve apontar para o próximo nó (ponterio “prox” do nó excluído);
- Liberar o nó excluído com `free()`;
- Não liberar o dado caso ele seja um ponteiro;