FaculdadeBarretos
www.faculdadebarretos.com.br

Algoritmos e Estruturas de Dados

(Aula 7 - Visão Geral de Estruturas Encadeadas Lineares)

Prof. Me. Diogo Tavares da Silva
contato: *diogotavares@unibarretos.com.br*

FaculdadeBarretos
www.faculdadebarretos.com.br

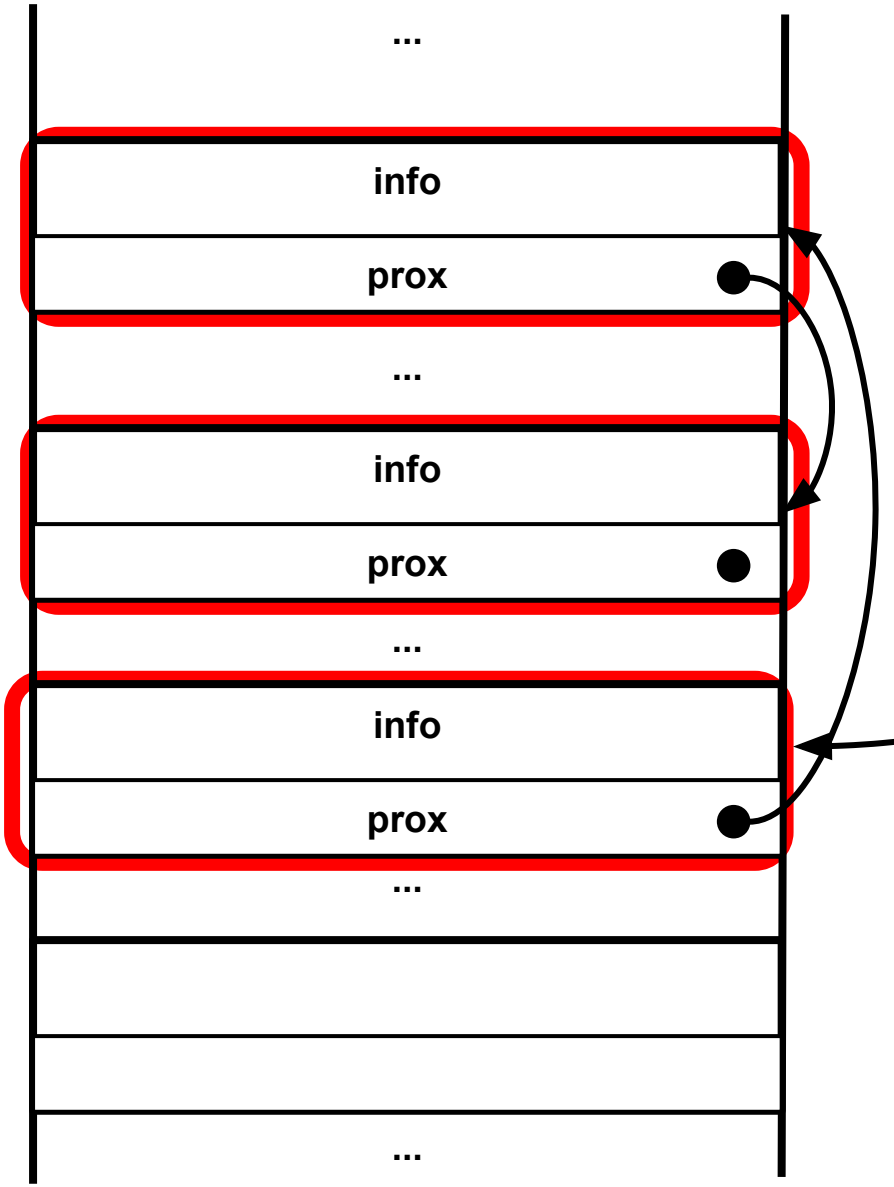
Nas últimas aulas aprendemos...

- Ponteiros e como usá-los para alterar o valor de variáveis e registros em funções
- Tipos estruturados utilizando o recurso de registros (***structs***)
- Como definir novos tipos de dados utilizando o *typedef*
- Por fim discutimos as vantagens e desvangens da alocação contínua de memória

A idéia de listas encadeadas

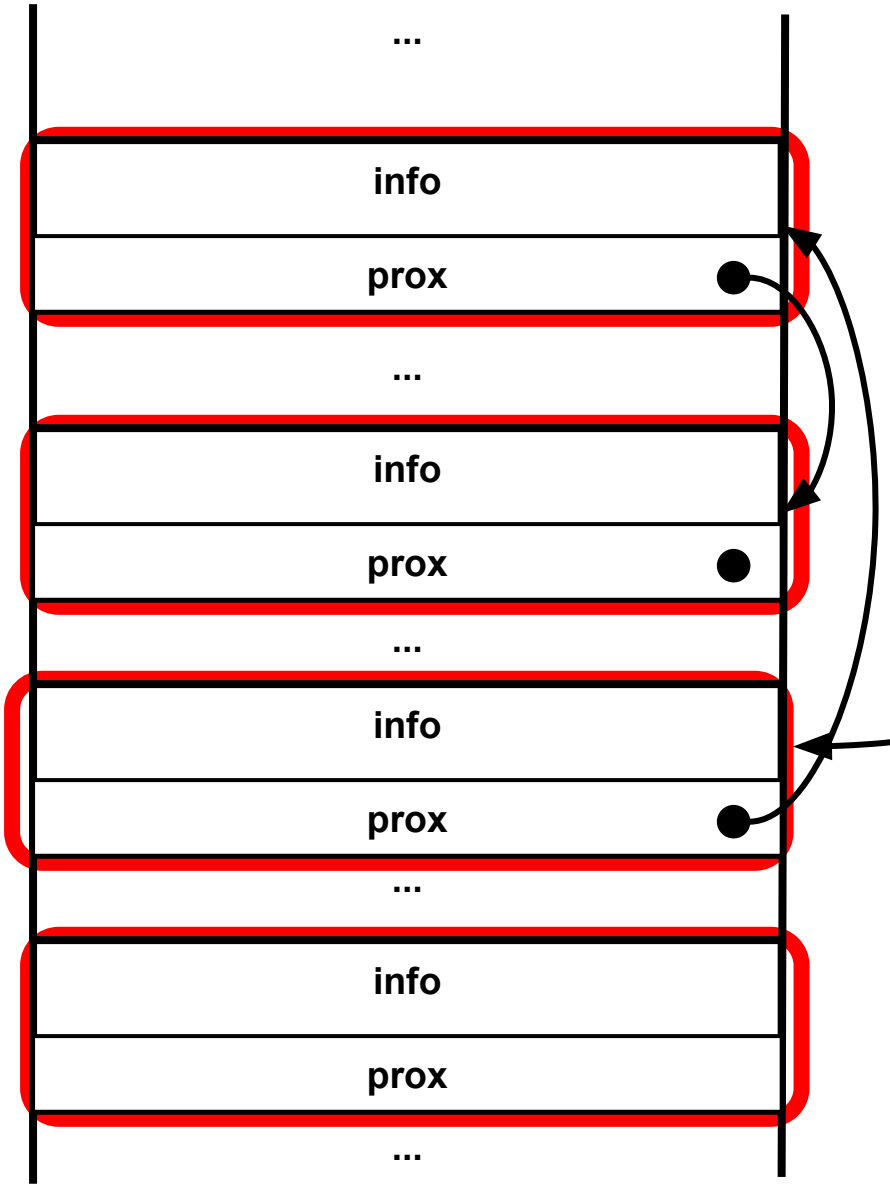
- Ao avaliar os problemas da alocação contínua, discutimos a necessidade de um padrão diferente de estrutura:
  - Uma estrutura dinâmica que pudesse ser alocada e desalocada conforme a necessidade em espaços não contíguos de memória.
  - Uma estrutura que fosse inserida unidade (nó) por unidade, ligando-as umas as outras.

A idéia de listas encadeadas



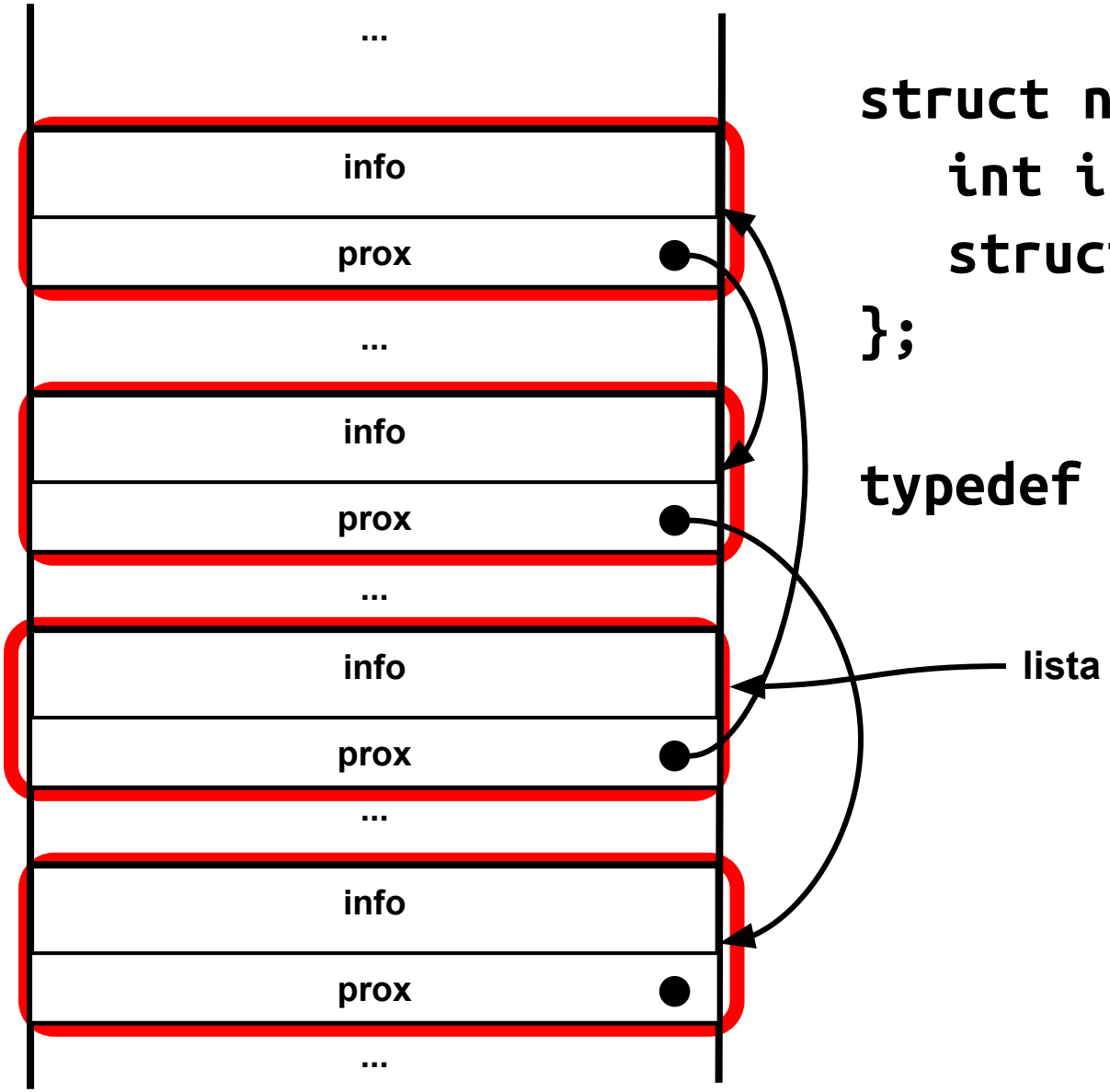

```
struct node{
   int info;
   struct node *prox;
};

typedef struct node NODE;
```

A idéia de listas encadeadas



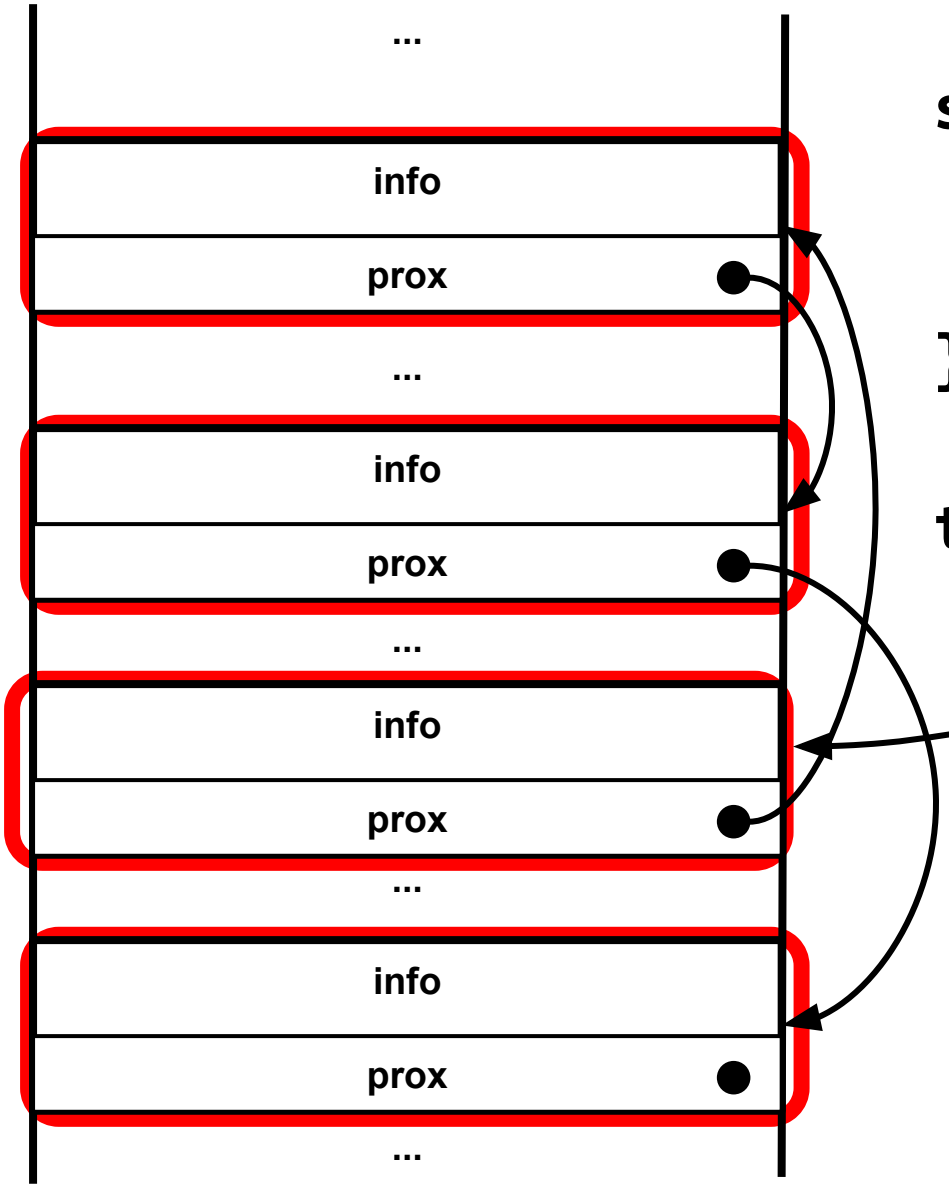

```
struct node{
   int info;
   struct node *prox;
};

typedef struct node NODE;
```

A idéia de listas encadeadas

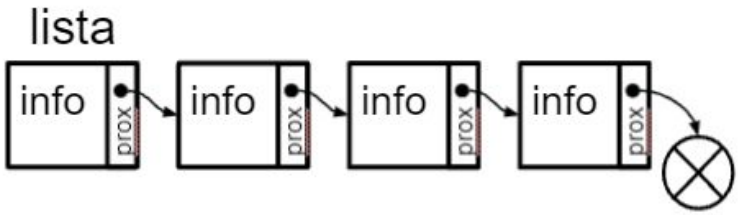



```
struct node{
   int info;
   struct node *prox;
};

typedef struct node NODE;
```

A idéia de listas encadeadas




```
struct node{
   int info;
   struct node *prox;
};

typedef struct node NODE;
```



Criação do tipo lista

- A partir da criação do elemento unitário (**node**), podemos criar a lista por duas abordagens:
  - **Abordagem 1:**
    - Criação de um tipo **LISTA** a partir do tipo **NODE**

```
struct lista{
   NODE *primElem; //ponteiro p/ primeiro elemento
   //outras varáveis que julgar relevantes
}
```

Criação do tipo lista

- A partir da criação do elemento unitário (**node**), podemos criar a lista por duas abordagens:
  - **Abordagem 2:**
    - MAIS COMUM!
    - Referenciamos a lista por meio de um ponteiro para o primeiro elemento (assim como o vetor)
    - Mantemos sempre esse ponteiro apontado para o primeiro elemento (Pra não perder o início da lista)

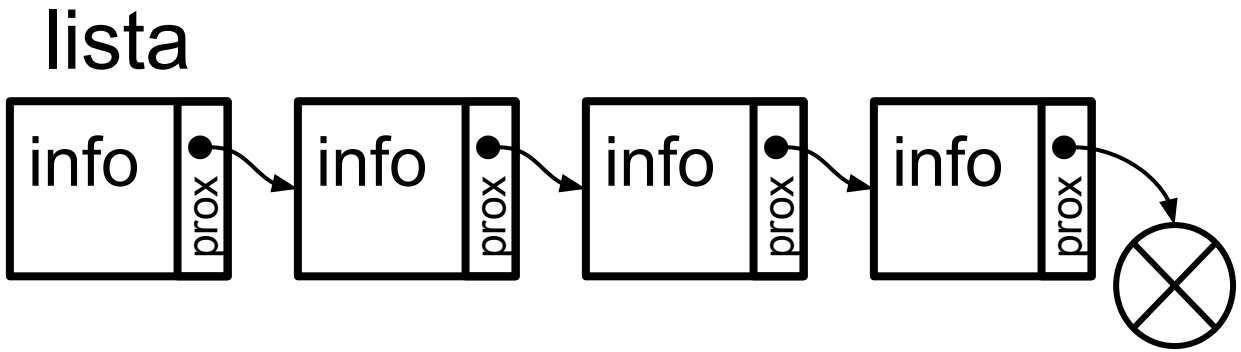
```
NODE* lista;
```

Criação do tipo lista

- A partir daí desenvolvemos todos os algoritmos de manipulação pra esse TAD (Tipo de dado abstrato)
  - Inserção
  - Busca
  - Remoção
  - Percurso

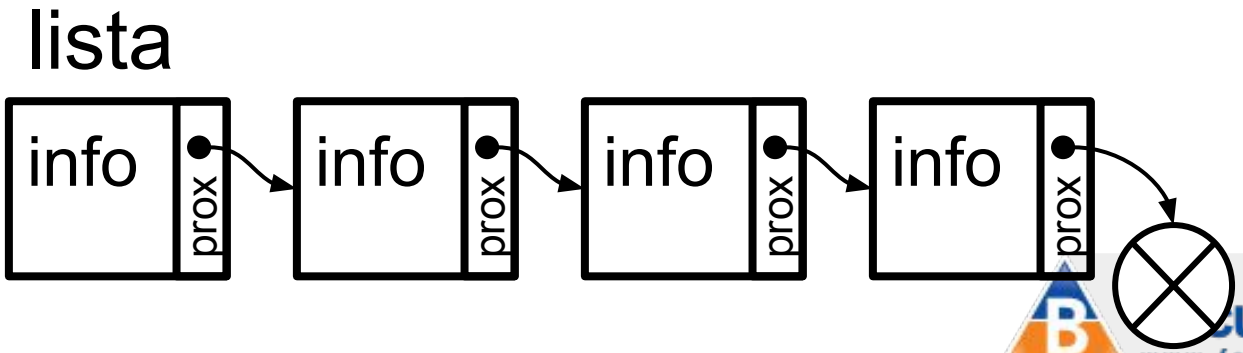
Tipos de estruturas de lista encadeada

- **Lista simplesmente encadeada**
  - A implementação mais clássica do tipo lista encadeada é a “lista simples”, como apresentada abaixo:



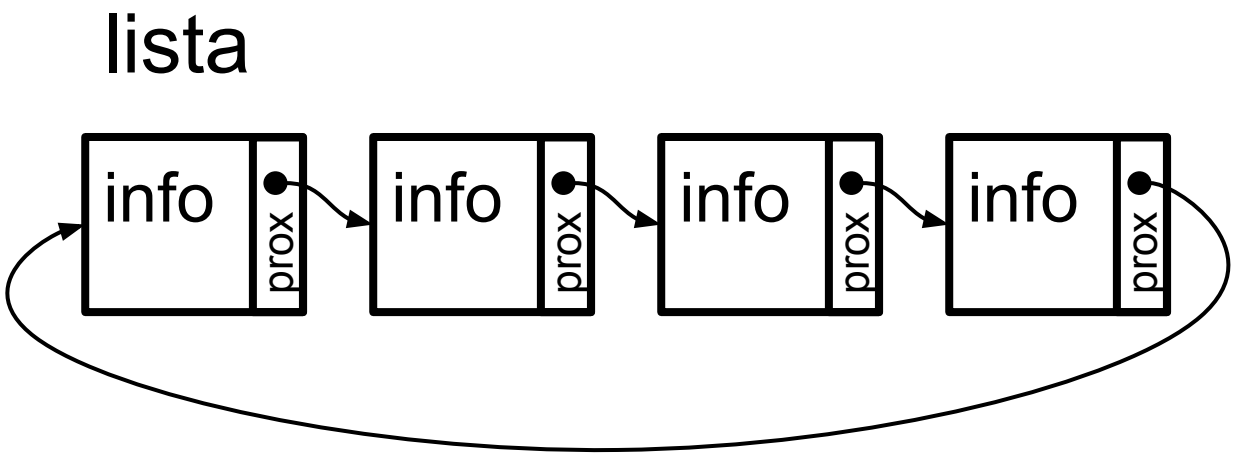
Tipos de estruturas de lista encadeada

- **Lista simplesmente encadeada**
  - ponteiro pro primeiro nó
  - apenas um encadeamento em uma única direção
  - uso de ponteiro auxiliar pra percorrer
    - não pode perder o começo da lista



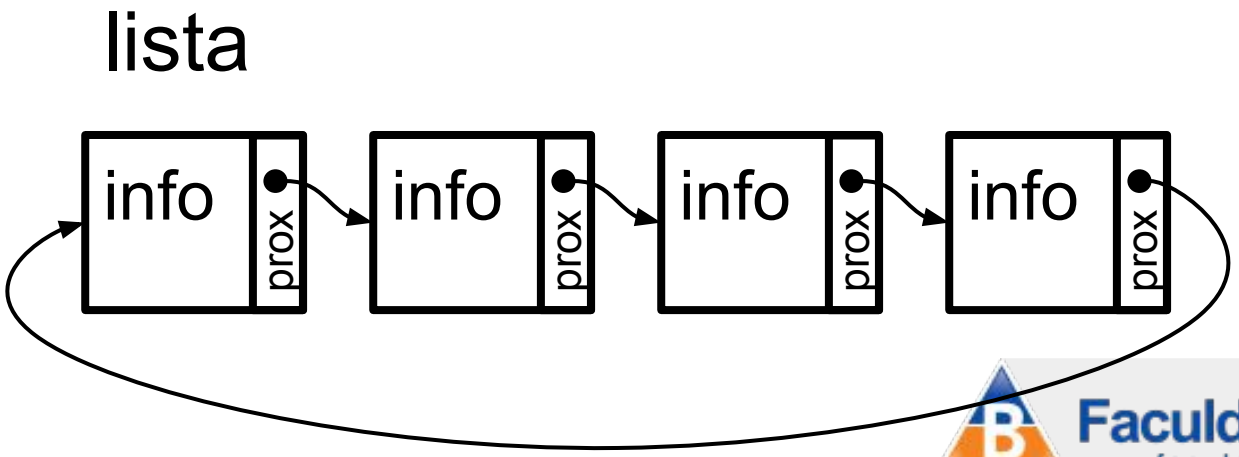
Tipos de estruturas de lista encadeada

- **Lista circular**



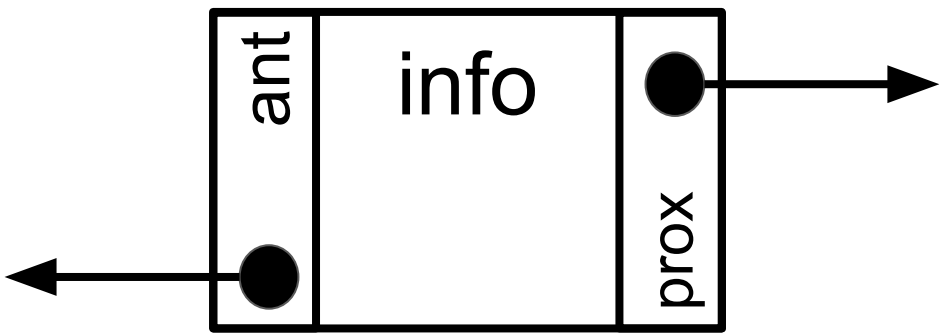
Tipos de estruturas de lista encadeada

- **Lista circular**
  - Encadeamento simples
  - encadeamento em anel unidirecional
  - Se a informação não tem ordem, cabeça da lista pode ficar em qualquer posição
    - útil para resolver problemas de “vez”



Tipos de estruturas de lista encadeada

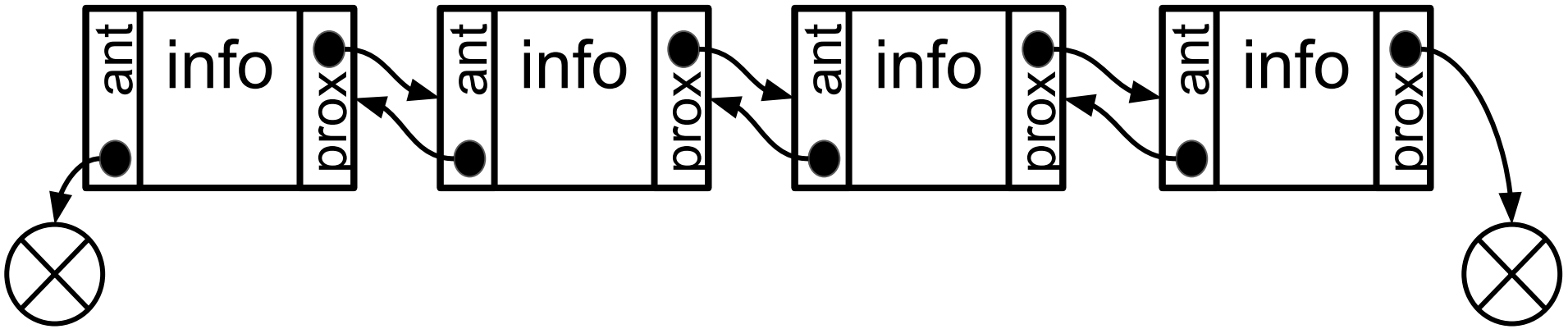
- **Lista duplamente encadeada**
  - Neste variante cada nó tem dois ponteiros ao invés de um
    - um ponteiro para o próximo nó
    - um ponteiro para o nó anterior




```
struct nodeDup{
   int info;
   struct nodeDup* ant;
   struct nodeDup* prox;
};
```

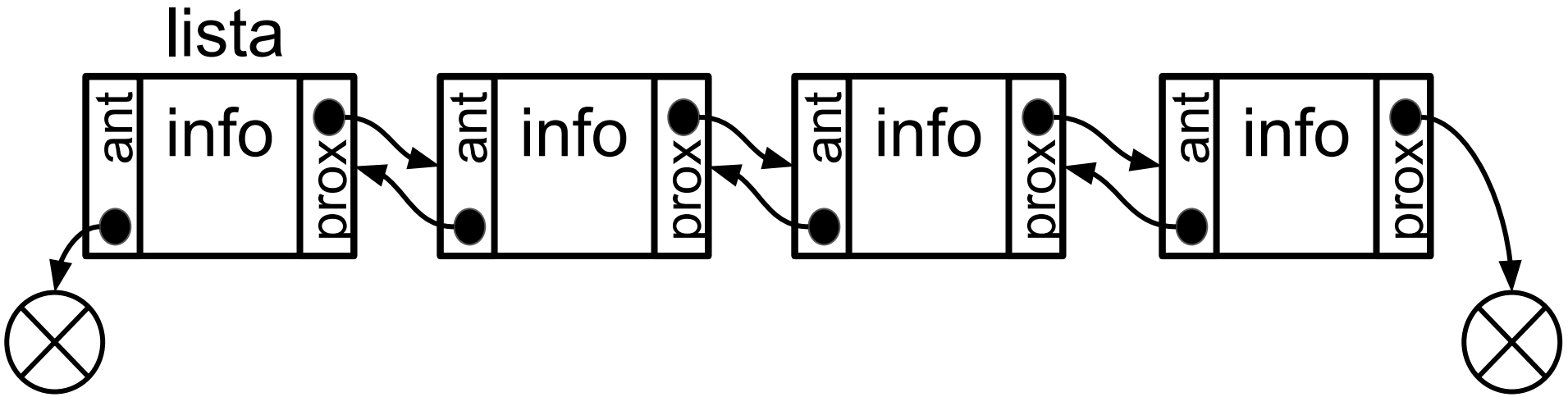
Tipos de estruturas de lista encadeada

- **Lista duplamente encadeada**



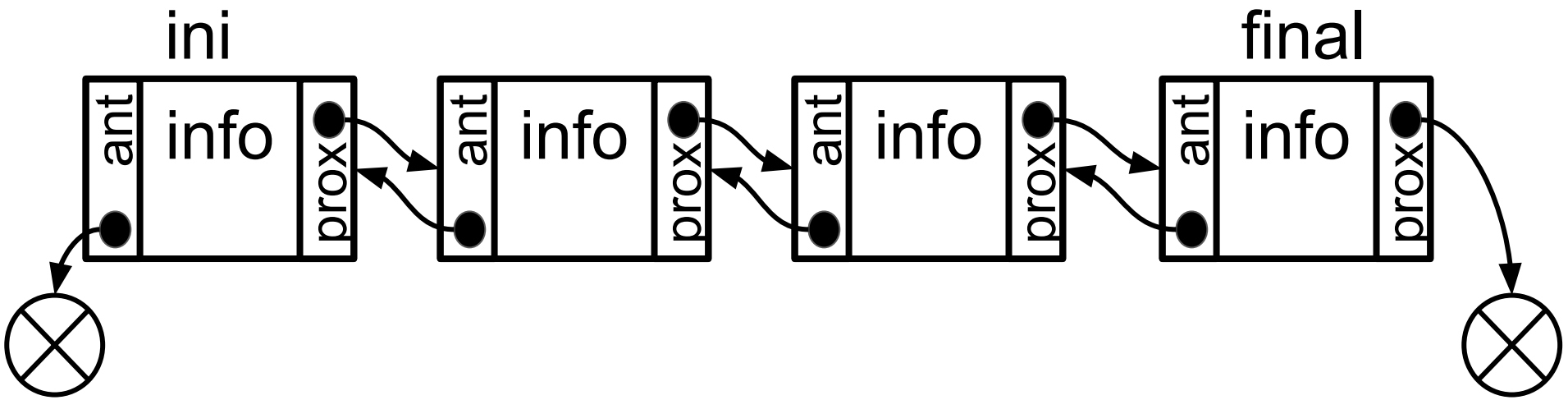
Tipos de estruturas de lista encadeada

- **Lista duplamente encadeada**
  - Controle de acesso aos nós:
    - duas abordagens:
      - ponteiro único



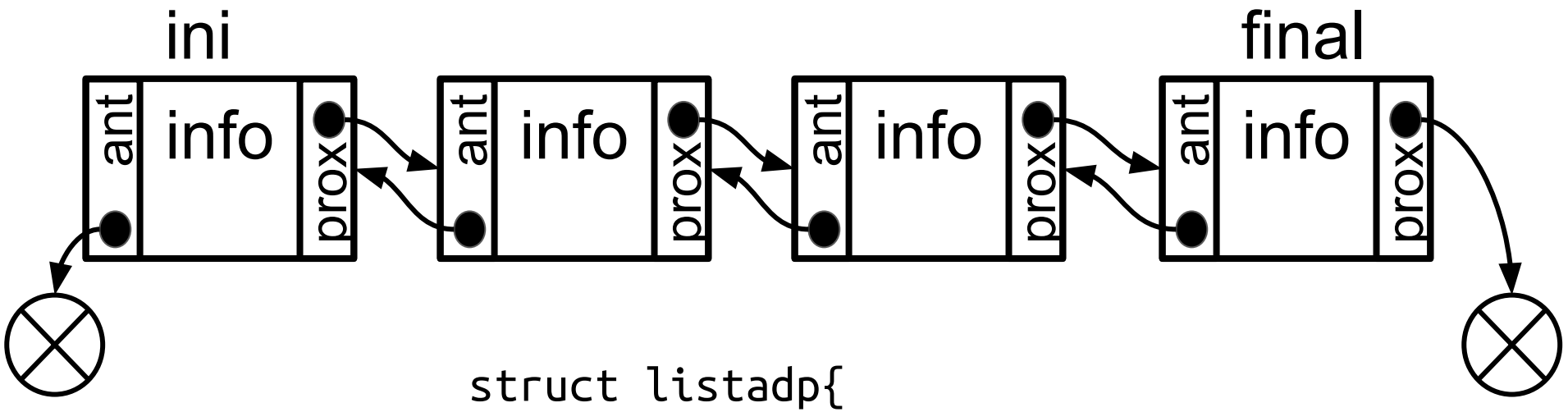
Tipos de estruturas de lista encadeada

- **Lista duplamente encadeada**
  - Controle de acesso aos nós:
    - duas abordagens:
      - estrutura com ponteiros pro início e final



Tipos de estruturas de lista encadeada

- **Lista duplamente encadeada**
  - Controle de acesso aos nós:
    - duas abordagens:
      - estrutura com ponteiros pro início e final




```
struct listadp{
    NODEDP* ini;
    NODEDP* final;
};
typedef struct listadp LISTADP;
```

Tipos de estruturas de lista encadeada

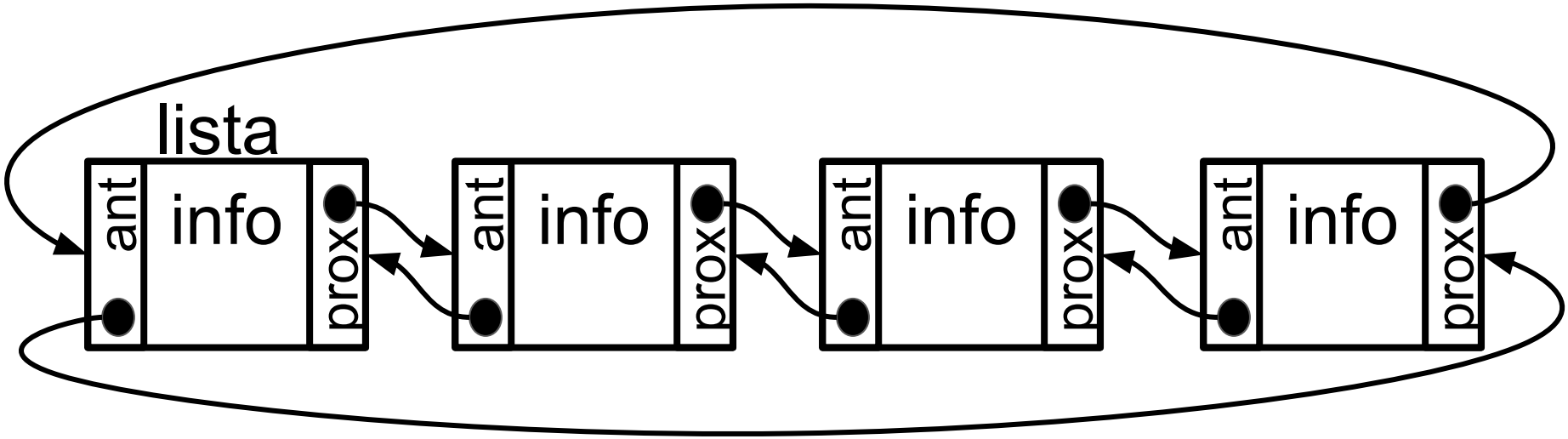
- **Lista duplamente encadeada**
  - encadeamento duplo bidirecional
  - Possibilidade de ponteiro no início e no fim
    - não é necessário percorrer pra encontrar o fim da estrutura
  - Possibilidade de percorrer a estrutura nas duas direções

Tipos de estruturas de lista encadeada

- **Lista duplamente encadeada**
  - **Porque não usar sempre listas duplamente encadeadas**
    - Nem sempre referenciar o anterior é necessário
    - Custo de implementação
      - número de ponteiros a mais armazenados?
  - **Quando usar?**
    - Quando as operações que minha aplicação irá utilizar custarão menos computacionalmente se implementadas com lista duplamente encadeada

Tipos de estruturas de lista encadeada

- **Lista circular duplamente encadeada**
  - Junção do conceito de listas circulares e listas duplamente encadeadas
    - Não faz mais sentido armazenar um ponteiro para o início e para o final
      - Um nó está ligado do outro



TADs unidimensionais criadas a partir de listas encadeadas

- A partir dessas possibilidades de implementações de listas encadeadas apresentadas
- podemos criar métodos específicos para definir comportamentos de acesso, inserção, busca e remoção úteis para resolver uma série de problemas
  - **disciplinas de acesso**

TADs unidimensionais criadas a partir de listas encadeadas

- **disciplina de acesso**
  - regras definidas de como inserimos, percorremos e buscamos e removemos elementos em uma estrutura encadeada
  - São importantes por diferenciar dois tipos principais de estruturas de dados
    - **Pilhas**
    - **Filas**

TADs unidimensionais criadas a partir de listas encadeadas

- **Pilha (*stack*)**
  - estrutura encadeada com disciplina de acesso **LIFO**
    - ***Last In - First out***
      - Ou seja, o último elemento inserido é o primeiro a ser removido
    - na prática:
      - Inserção e remoção de elemento ocorrem de apenas um lado da lista encadeada
        - começo da lista
      - é a estrutura de dados encadeada mais simples possível

TADs unidimensionais criadas a partir de listas encadeadas

- **Pilha (*stack*)**
  - Utilizações
    - **Manter o histórico de inserções e remoções original**
      - pilha de processos na memória
      - desfazer e refazer
      - avançar e retroceder

TADs unidimensionais criadas a partir de listas encadeadas

- **Fila (*queue*)**
  - estrutura encadeada com disciplina de acesso **FIFO**
    - ***First In - First out***
      - Ou seja, o primeiro elemento a ser inserido é o primeiro a ser removido
    - na prática:
      - Inserção por um lado da estrutura e remoção pelo outro lado
      - conveniente usar lista dupla e manter ponteiro apontando pro início e final da fila

TADs unidimensionais criadas a partir de listas encadeadas

- **Fila (*queue*)**
  - Utilizações
    - Problemas em que estamos tentando manter uma ordem de acesso e atendimento de casos
      - análogo aos problemas de filas reais

TADs unidimensionais criadas a partir de listas encadeadas

- **Fila (*queue*)**
  - variações
    - Fila circular
    - Fila dupla (*deque*)