Estrutura de Dados

Aula 05 – Lista duplamente encadeada e circular

Prof. Dr. Daniel Vecchiato

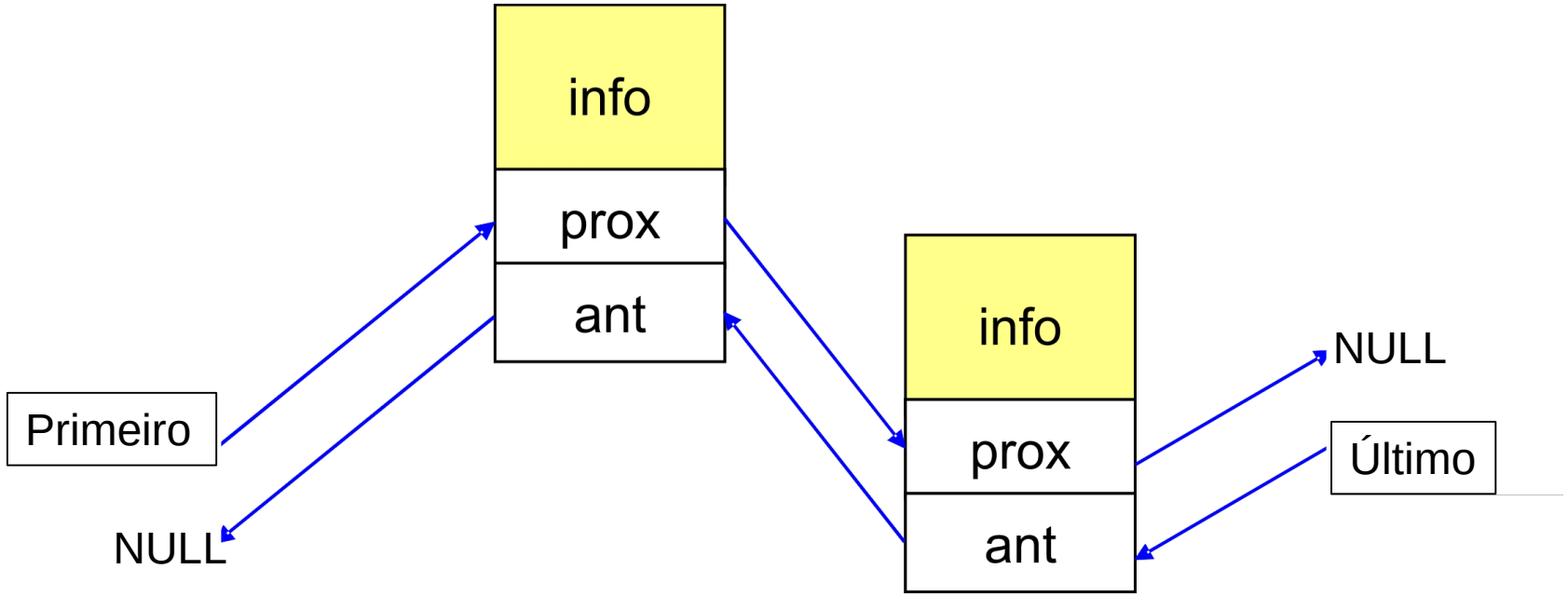
Agenda

- Introdução
- Lista duplamente encadeada
- Lista circular
- Exercícios

Introdução

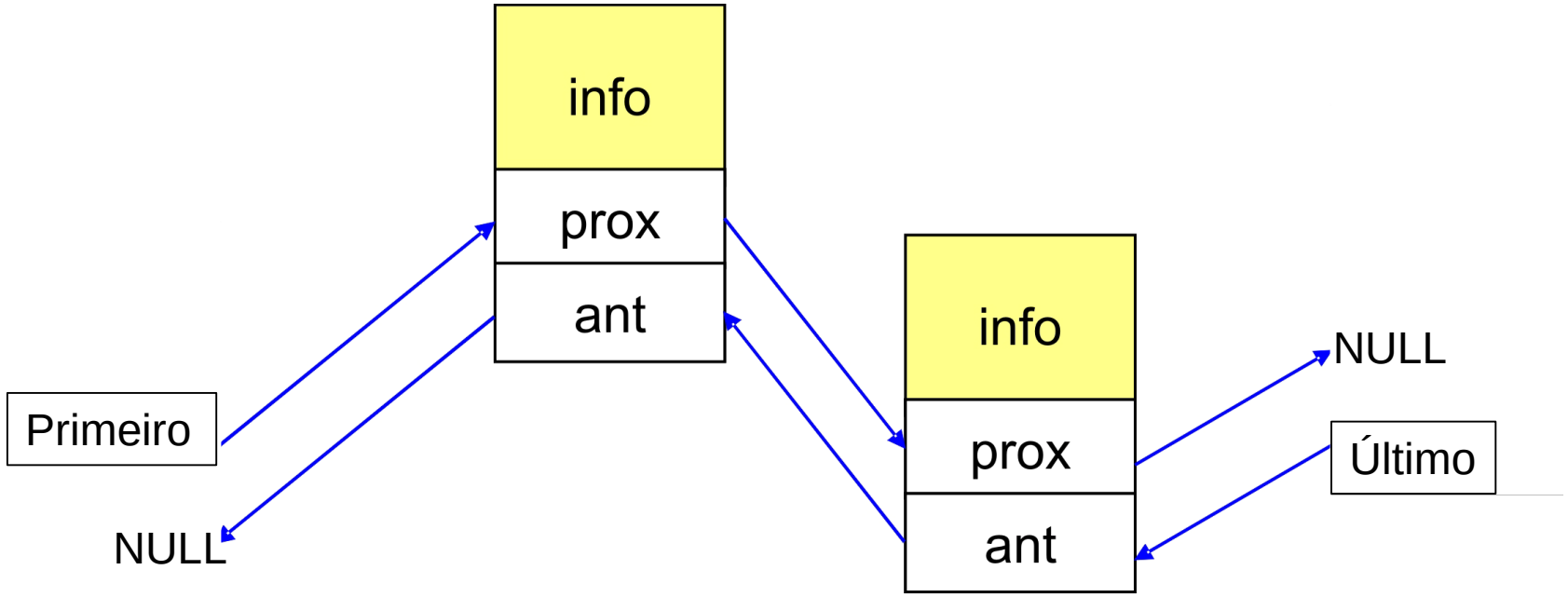
- Alguns tipos de listas:
  - Lista estática
  - Lista dinâmica
    - Lista simplesmente encadeada
      - Com cabeça
      - Sem cabeça
    - Lista duplamente encadeada
    - Lista circular

Lista duplamente encadeada



Lista duplamente encadeada

- Cada elemento aponta para os elementos anterior e posterior à ele
  - Dado um elemento, é possível acessar o próximo e o anteiror
  - Se houver um ponteiro para o último elemento, é possível percorrer a lista na ordem inversa
  - Melhor desempenho quando há muitas inserções e remoções de elementos intermediários



Lista duplamente encadeada

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int chave;
} TItem;

typedef struct celula {
    struct celula *pAnt;
    struct celula *pProx;
    TItem item;
} TCelula;

typedef struct {
    TCelula *pPrimeiro, *pUltimo;
} TLista;
```

Lista duplamente encadeada

```
void iniciarLista (TLista *pLista);
int isVazia (TLista *pLista);
int inserir (TLista *pLista, TItem x);
int removerPrimeiro (TLista *pLista, TItem *pX);
void imprimir (TLista *pLista, int inverso);
```

Lista duplamente encadeada

```
void iniciarLista (TLista *pLista) {
    pLista->pPrimeiro = NULL;
    pLista->pUltimo = NULL;
}

int isVazia (TLista *pLista) {
    return pLista->pPrimeiro == NULL;
}
```

Lista duplamente encadeada

```
int inserir (TLista *pLista, TItem x) {
    TCelula *novo = (TCelula *) malloc (sizeof (TCelula));
    novo->item = x;
    novo->pAnt = NULL;
    novo->pProx = NULL;

    if (isVazia (pLista)) {
        pLista->pPrimeiro = novo;
        pLista->pUltimo = novo;
    } else {
        pLista->pUltimo->pProx = novo;
        novo->pAnt = pLista->pUltimo;
        pLista->pUltimo = novo;
    }
    return 1;
}
```

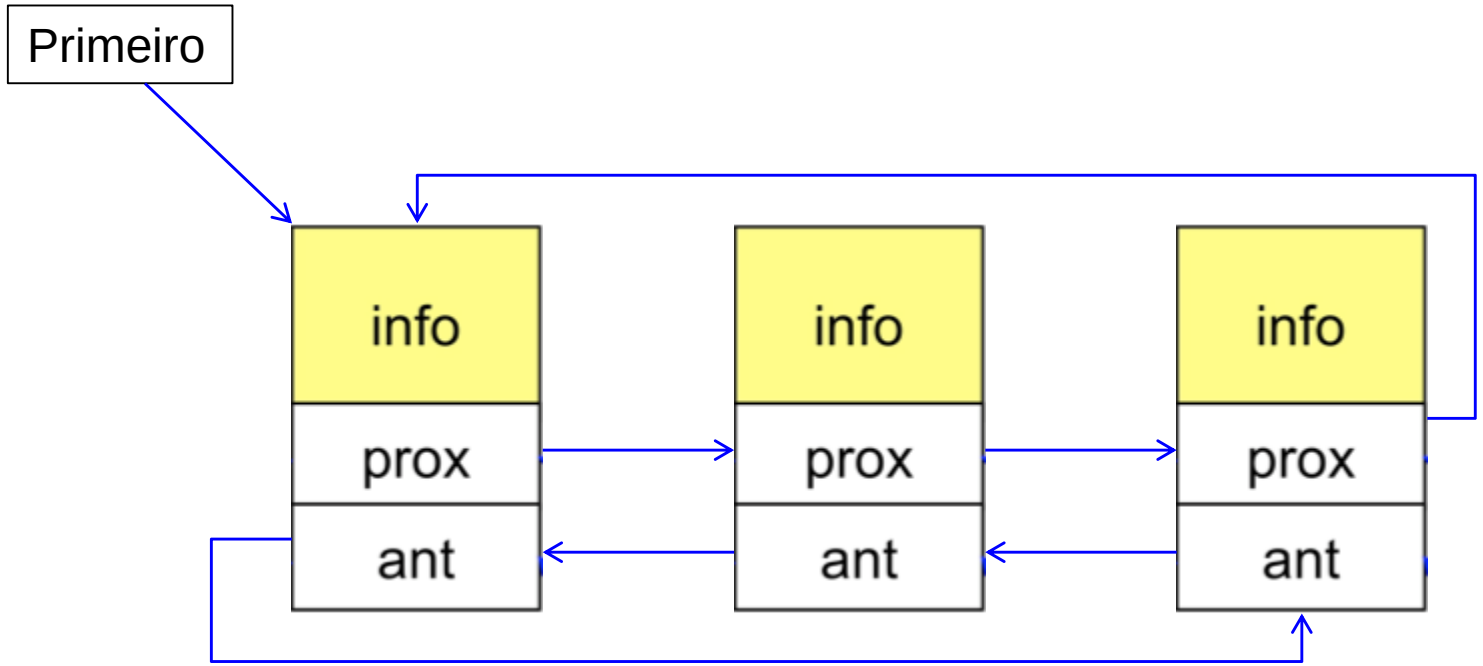
Lista duplamente encadeada

```
int removerPrimeiro (TLista *pLista, TItem *pX) {
    if (isVazia (pLista))
        return 0;
    TCelula *pAux;
    pAux = pLista->pPrimeiro;
    *pX = pAux->item;
    pLista->pPrimeiro = pAux->pProx;
    pLista->pPrimeiro->pAnt = NULL;
    free (pAux);
    return 1;
}
```

Lista duplamente encadeada

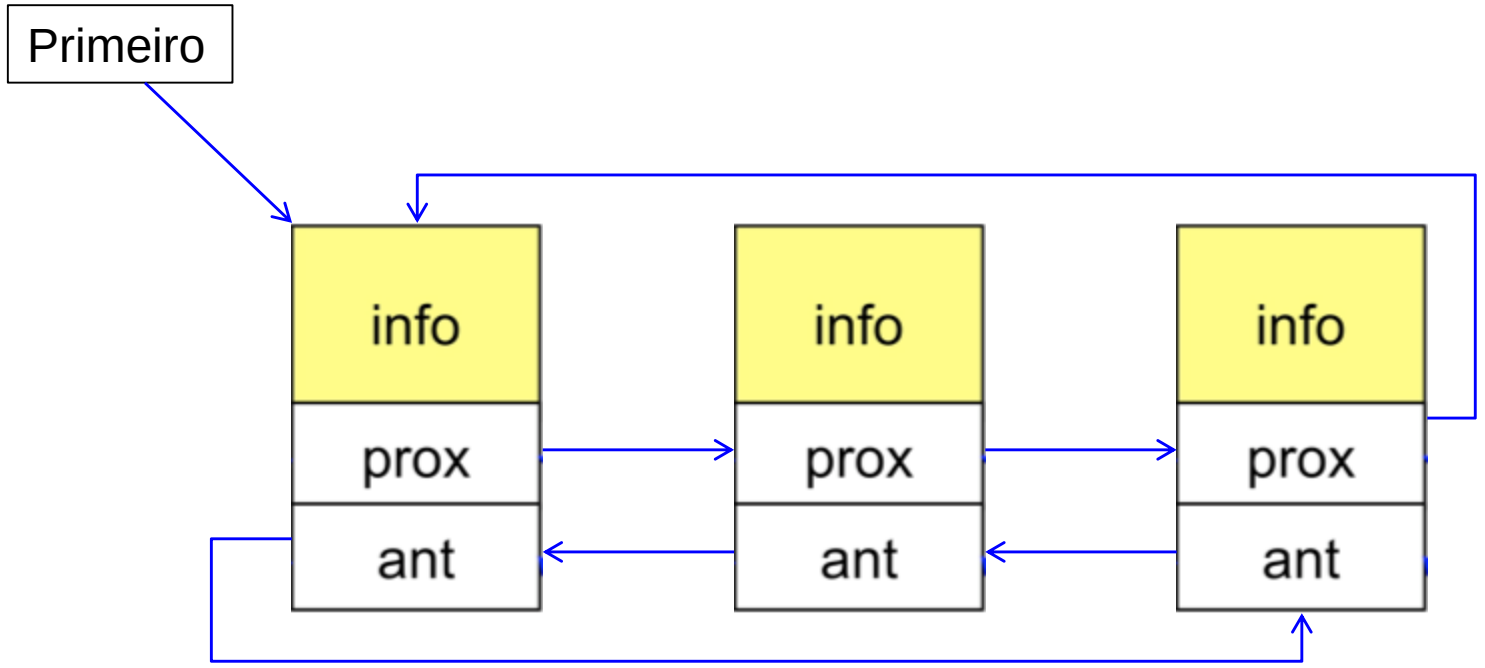
```
void imprimir (TLista *pLista, int inverso) {
    TCelula *celula;
    printf("Itens da lista: ");
    if (inverso) {
        celula = pLista->pUltimo;
    } else {
        celula = pLista->pPrimeiro;
    }
    while (celula != NULL) {
        printf("%d ", celula->item.chave);
        if (inverso) {
            celula = celula->pAnt;
        } else {
            celula = celula->pProx;
        }
    }
    printf("\n");
}
```

Lista circular



Lista circular

- O primeiro elemento aponta para o último e vice-versa.
- Ainda há a marcação da onde está o início da lista
- Não há necessidade de marcar o fim da lista



Lista circular

```
void imprimir (TLista *pLista) {
    TCelula *celula = pLista->pPrimeiro;

    printf("Itens da lista: ");

    if (celula != NULL) {
        do {

            printf("%d ", celula->item.chave);
            celula = celula->pProx;

        } while (celula != pLista->pPrimeiro);
    }

    printf("\n");
}
```

Exercícios

- Implemente as duas listas: duplamente encadeada e circular
- Implemente uma função de busca na lista duplamente encadeada

```
TCelula* busca (TLista *pLista, int chave) {
  ...
}
```

  - pLista: lista a ser pesquisada
  - chave: informação a ser encontrada dentro de uma célula
  - Retorna o ponteiro de uma célula que contém a chave pesquisada ou NULL caso a chave não exista na lista
- Implemente uma função para remover qualquer elemento da lista, independente de sua posição, na lista circular

```
int remover (TLista *pLista, int chave) {
  ...
}
```

  - Os elementos a serem removidos devem conter a chave passada por parâmetro
  - Retorna quantos elementos foram removidos

Estrutura de Dados

Material elaborado por:
Thiago Meirelles Ventura

Baseado em:

- Ascencio, A. F. G; Araújo, G. S. Estruturas de Dados. Pearson, 2011.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. Algoritmos: teoria e prática. Elsevier, 2002.
- Aulas do Prof. Reinaldo Silva Fortes (http://www.decom.ufop.br/reinaldo/)