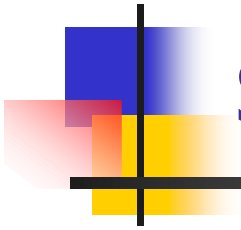


SIN211 Algoritmos e Estruturas de Dados

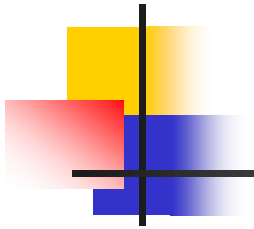
Prof. João Batista Ribeiro

joao42lbatista@gmail.com

UFV

Universidade Federal de Viçosa

Slides baseados no material da Prof.ª Rachel Reis

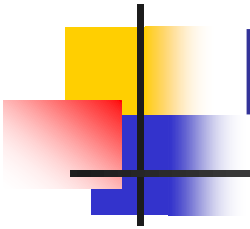


Aula de Hoje

- Lista Duplamente Encadeada

Situação-Problema

- É comum no processo inicial de alfabetização de crianças canhotas, elas escreverem na folha do caderno começando pelo lado direito em direção ao esquerdo. Esse processo, quando acompanhado pelos pais e professores, é corrigido logo no início da alfabetização. Considerando que Joãozinho é canhoto e se encontra nessa fase, apresente uma solução usando o conceito de lista que armazene o texto que Joãozinho escreveu (invertido) e imprima na ordem correta.



Listas Encadeadas

- Classificação:
  - Lista Simplesmente Encadeada
  - Lista Duplamente Encadeada

Revisando Lista Simplesmente Encadeada

- Um nó em uma lista simplesmente encadeada possui basicamente dois itens:
  - informação
  - ponteiro para o próximo nó
- Limitações:
  - Não permite percorrer a lista na ordem inversa
  - O processo de remoção exige a presença de um ponteiro auxiliar para acessar o nó anterior ao que desejamos remover.

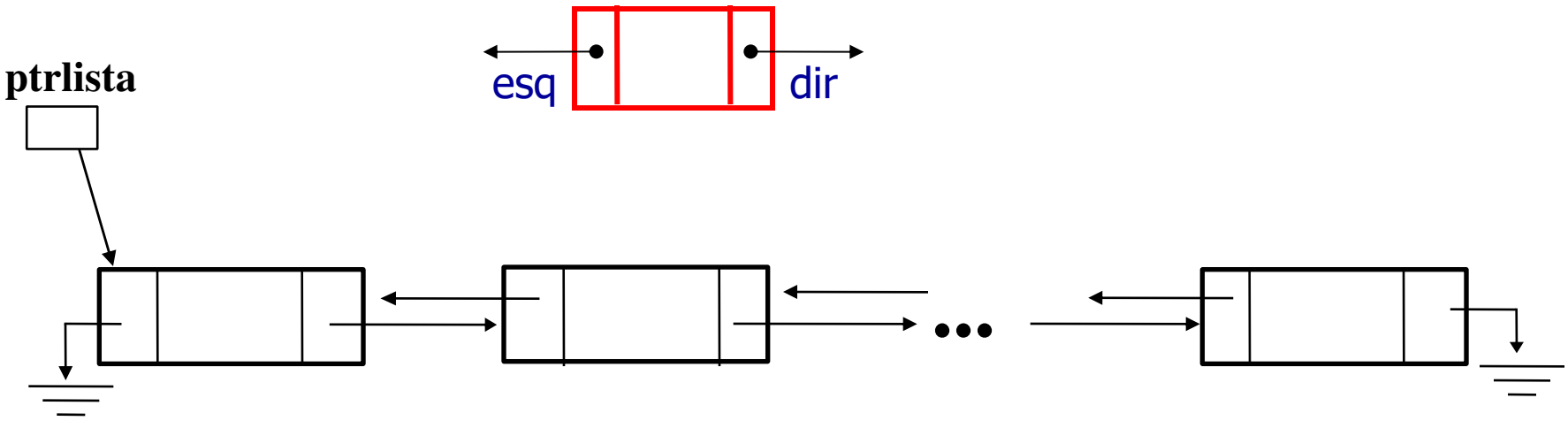
Revisando listas simplesmente encadeadas

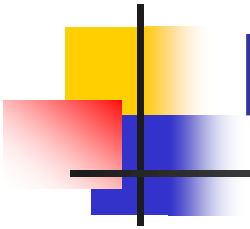
- Definição:

```
typedef struct sCELULA {
    <TIPO> info;
    struct sCELULA* next;
}CELULA;
```

Lista Duplamente Encadeada

- Cada nó em uma lista duplamente encadeada possui dois ponteiros, um para seu predecessor (ou nó à esquerda) e outro para seu sucessor (ou nó à direita):





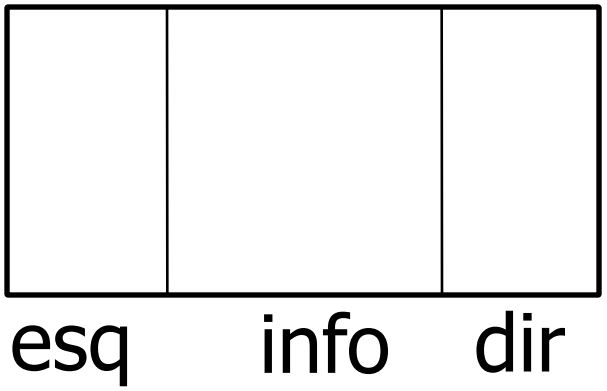
Lista Duplamente Encadeada

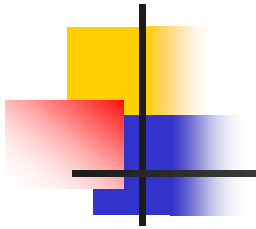
- Vantagens
  - A partir de um nó é possível acessar os nós adjacentes: direito e esquerdo
  - É possível remover um elemento da lista conhecendo apenas o endereço do nó
  - Facilita o processo de inserção à direita e à esquerda de um nó
  - Permite que a lista seja percorrida em ambas as direções.
- Desvantagem
  - Maior gasto de memória

Lista Duplamente Encadeada

- Podemos considerar os nós de uma lista duplamente encadeada consistindo de 3 campos:
  - o campo **info** contém as informações armazenadas no nó;
  - os campos **esq** e **dir**, que contém ponteiros para os nós de ambos os lados.

```
typedef struct sCELULA {
    <TIPO> info;
    struct sCELULA* esq;
    struct sCELULA* dir;
}CELULA;
```





Lista Duplamente Encadeada

- Operações
  - → inserção
  - → remoção

Lista Duplamente Encadeada (implementação)

- Insere elemento no início da lista

```
void insere_inicio(CELULA **lista, int x){
    CELULA *q;
    q = getnode ();
    if (q != NULL) {
        q->info = x;
        q->esq = NULL;
        q->dir = *lista;
        if (!empty(*lista))
            (*lista)-> esq = q;
        *lista = q;
    } else {
        printf ("\nERRO: falha na alocacao do noh.\n");
    }
}
```

Lista Duplamente Encadeada (implementação)

- Inserir no final da lista

```
void insere_fim (CELULA **lista, int x){
    CELULA *q;
    CELULA *aux;

    q = getnode ();
    if (q != NULL) {
        q->info = x;
        q->esq = NULL;
        q->dir = NULL;

        if (empty(*lista))
            *lista = q;
        ...
```

Lista Duplamente Encadeada (implementação)

- Inserir no final da lista (cont.)

```
        else { // percorre lista até chegar ao ultimo nó
           aux = *lista;
           while (aux->dir != NULL)
                aux = aux->dir;

           aux->dir = q;
           q -> esq = aux;
        }
    } else { // Fim do if(q != NULL)
        printf ("\nERRO na alocação do nó.\n");
    }
}
```

Lista Duplamente Encadeada (implementação)

- Remoção no início da lista

```
void remove_inicio(CELULA **lista){
    CELULA *q;

    q = *lista;
    if (!empty(*lista)) { // há itens na lista
        *lista = q->dir;
        if((*lista)!=NULL)
            (*lista)->esq = NULL;
        freenode (q);
    } else {
        printf ("\nERRO: lista vazia.\n");
    }
}
```

Lista Duplamente Encadeada (implementação)

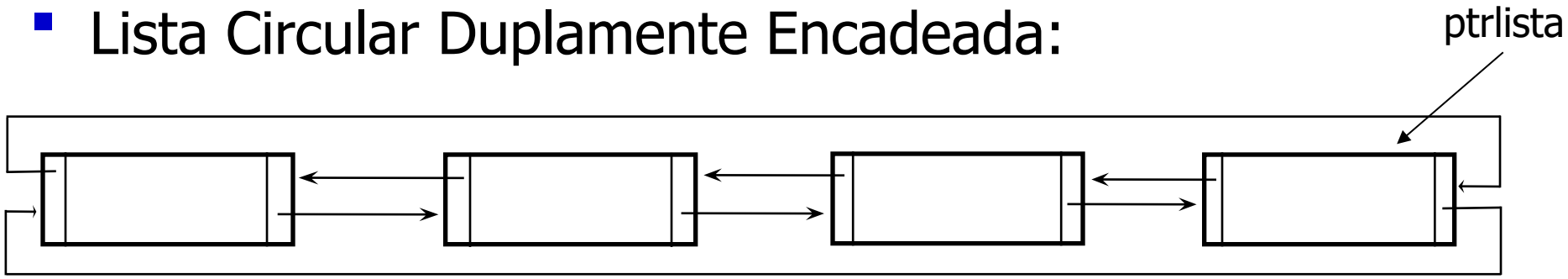
```
int remove_valor (CELULA **lista, int x) {
    CELULA *q;

    if ((q = pesquisa (*lista, x)) != NULL){
        if (*lista == q)  // nó está no início da lista
            remove_inicio (lista);
        else {
            (q->esq)->dir = q->dir;
            if(q->dir!=NULL)
                (q->dir)->esq = q->esq;
            freenode (q);
        }
        return 1; // removeu
    }
    return 0; //não removeu
}
```

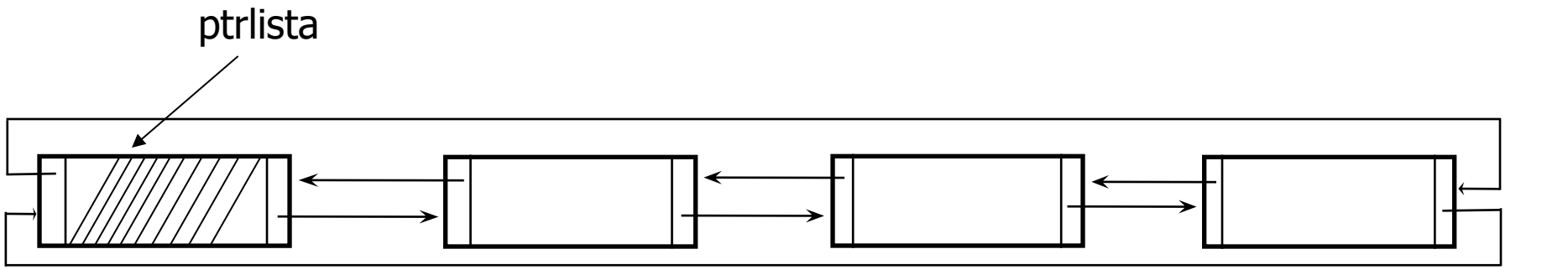
Remoção de um elemento x da lista

Lista Duplamente Encadeada

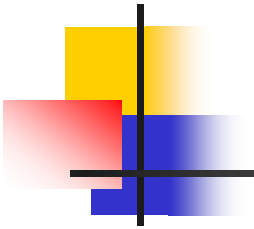
- Lista Circular Duplamente Encadeada:



- Lista Circular Duplamente Encadeada com cabeçalho



Nó de cabeçalho



Exercício

1) Implemente uma solução usando a estrutura de dados Lista Duplamente Encadeada para o problema abaixo:

   É comum no processo inicial de alfabetização de crianças canhotas, elas escreverem na folha do caderno começando pelo lado direito em direção ao esquerdo. Esse processo, quando acompanhado pelos pais e professores, é corrigido logo no início da alfabetização. Considerando que Joãozinho é canhoto e se encontra nessa fase, apresente uma solução usando o conceito de lista que armazene o texto que Joãozinho escreveu (invertido) e imprima na ordem correta.

Exercício

2) Crie um arquivo ListaDuplamenteEncadeada.c e implementa as seguintes informações de uma lista duplamente encadeada:
   - ✓ Definição
   - ✓ Operações
     - init
     - getnode
     - freenode
     - empty
     - exibe_lista
     - insere_inicio, insere_fim
     - remove_inicio, remove_valor
     - pesquisa

→ Teste seu programa criando um menu de opções para as principais operação.

Leituras Recomendadas

- DROZDEK, Adam. Estrutura de Dados e Algoritmos em C++. Editora Pioneira Thomson Learning, 2005.

→Pág. 85, seção 3.3 (Listas Circulares) - até pág. 96

→ Pág. 80, seção 3.2 (Lista Duplamente Ligada)– até pág. 84

- TENENBAUM A., LANGSAM Y. e AUGENSTEIN M. J. Estrutura de Dados usando C. Editora Makron, 1995.

→Pág. 279, seção 4.5 (Lista Circular) – até pág. 280

→Pág. 294(Lista Duplamente Ligada) – até pág. 300

→ Pág. 287(Nós de Cabeçalho) – até pág. 291