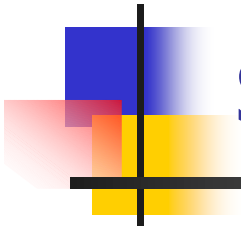


SIN211 Algoritmos e Estruturas de Dados

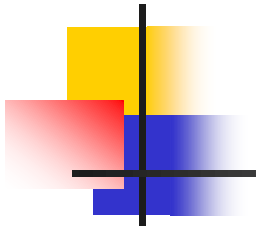
Prof. João Batista Ribeiro

joao42lbatista@gmail.com

UFV

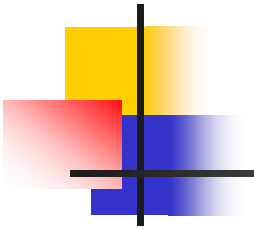
Universidade Federal de Viçosa

Slides baseados no material da Prof.ª Rachel Reis



Aula de Hoje

- Lista Circular



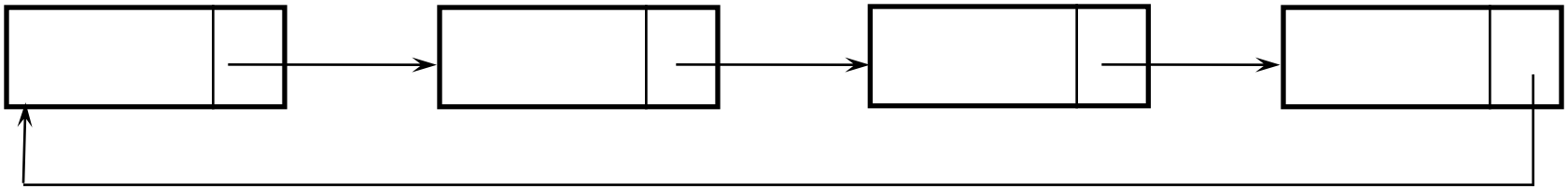
Situação-Problema

- Suponha que você precise desenvolver uma aplicação em que diversos processos de um sistema operacional utilizam concorrentemente um recurso (exemplo: processador). Além disso, é preciso garantir que nenhum processo acesse o recurso antes de todos os outros o utilizarem.

→ Descreva uma solução para o cenário descrito anteriormente usando o conceito de lista simplesmente encadeada.

Lista Encadeada Circular

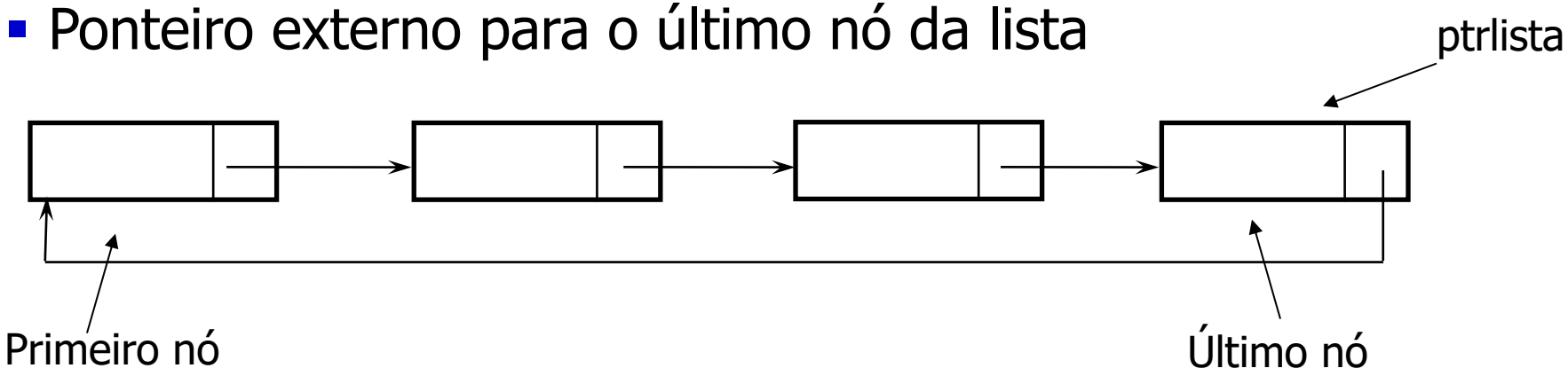
- Em uma lista circular os nós formam um anel.



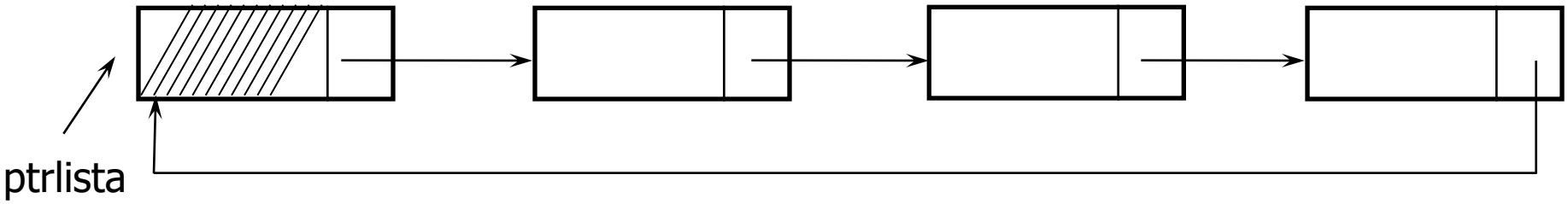
- O último elemento da lista tem como próximo elemento o primeiro elemento desta lista, formando um ciclo.

Lista Encadeada Circular
Possíveis implementações

- Ponteiro externo para o último nó da lista

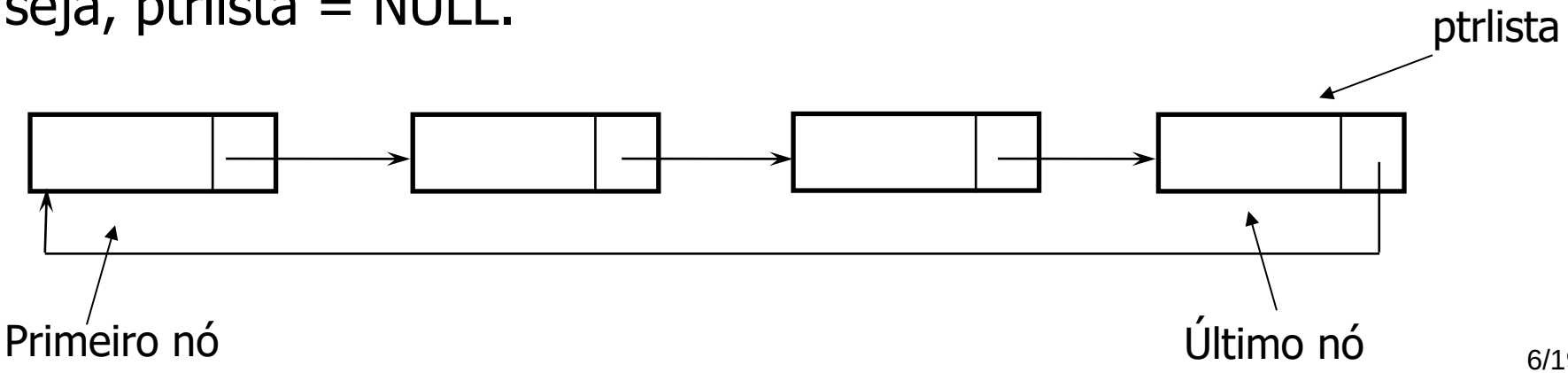


- Nó cabeçalho



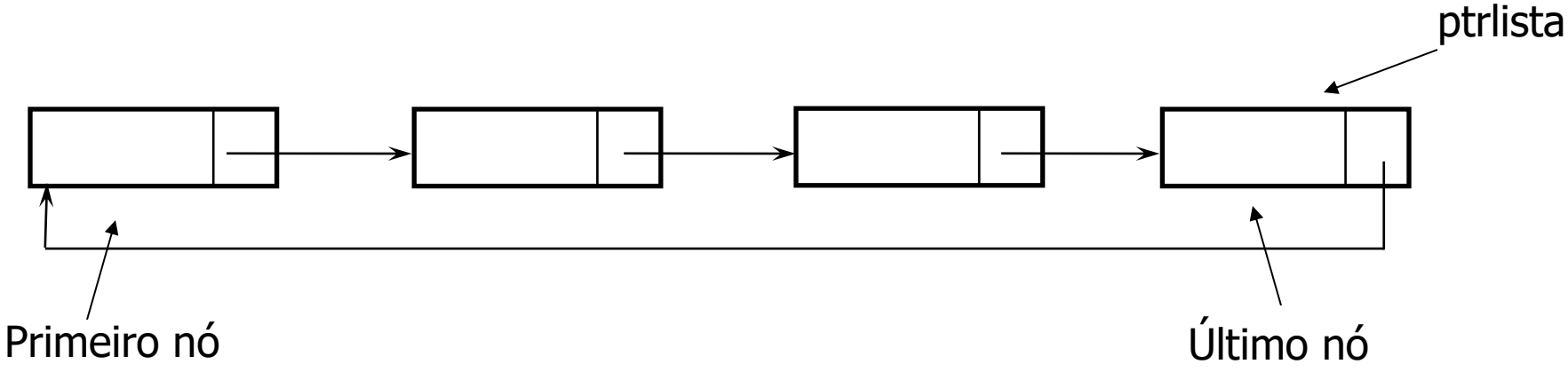
Lista Encadeada Circular

- Observe que a lista circular não tem o "primeiro" ou o "último" nó natural. Sendo assim, é preciso estabelecê-los por convenção.
- A convenção mais útil é permitir que o ponteiro externo para a lista circular aponte para o último nó, e que o nó seguinte se torne o primeiro nó:
- A lista vazia é representada pelo ponteiro externo nulo, ou seja, ptrlista = NULL.



Lista Encadeada Circular

- Se ptrlista é um ponteiro externo para uma lista circular, ptrlista referencia o último nó e ptrlista->next referencia o primeiro nó da lista.
- **Vantagem desta convenção:**
  - Poder incluir ou remover um elemento convenientemente a partir do início ou final de uma lista.



Lista Encadeada Circular - Implementação

- A estrutura de dados lista encadeada circular é definida da mesma forma que uma lista encadeada simples.

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

Lista Encadeada Circular - Implementação

- A estrutura de dados lista encadeada circular formada por elementos do tipo inteiro.

```
typedef struct sCELULA{
    int info;
    struct sCELULA* next;
}CELULA;
```

Lista Encadeada Circular - Implementação

```
void insere_fim(CELULA **lista, int x){
    CELULA *q;
    q = getnode();
    if (q != NULL) {
       q->info = x;
       if (empty(*lista)){
          q->next = q;
       } else{  //insere no fim da lista
           q->next = (*lista)->next;
       }    (*lista)->next = q;
       *lista = q;
    } else { // Fim do if(q != NULL)
        printf ("\nERRO na alocação do nó.\n");
    }
}
```

Insere elemento na lista cirular

Lista Encadeada Circular - Implementação

```
void listar (CELULA *lista) {
    CELULA *aux;

    if(!empty(lista)) {
        aux = lista->next;
        do{
          printf ("%d\t", aux->info);
          aux = aux->next;
        }while(aux != lista->next);
    } else {
        printf("\nNao ha elemento na lista.");
    }
    printf("\n");
}
```

Exibe elementos da lista circular

Lista Encadeada Circular - Implementação

```
void remove_inicio (CELULA **lista){
    CELULA *aux;
    if (!empty(*lista)) {  //há itens na lista
       if ((*lista) == (*lista)->next){
           freenode(*lista);
           *lista = NULL;
       }else{
           aux = (*lista)->next;
           (*lista)->next = aux->next;
           freenode(aux);
       }
    }else{
        printf ("\nERRO: lista vazia.\n");
    }
}
```

Remove elementos da lista circular

Lista Encadeada Circular - Implementação

- Função main

```
int main(){
    CELULA *ptrlista;
    init(&ptrlista);

    insere_fim(&ptrlista, 9);
    insere_fim(&ptrlista, 1);
    insere_fim(&ptrlista, 3);

    listar(ptrlista);
    remove_inicio (&ptrlista);
    listar(ptrlista);
    remove_inicio (&ptrlista);
    listar(ptrlista);
}
```

Nó de cabeçalho

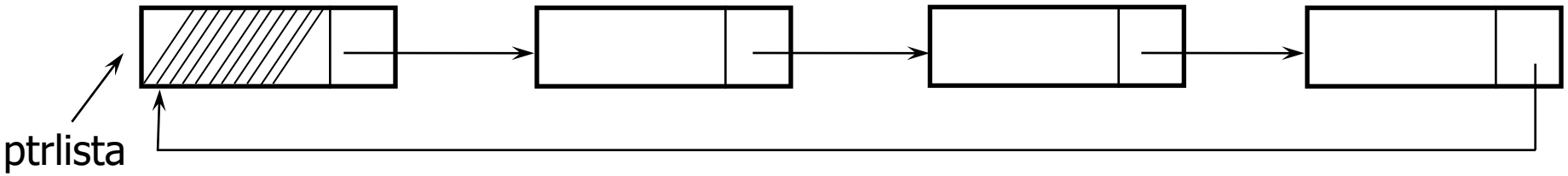
- Suponha que precisemos percorrer uma lista circular

- Executar *aux = aux → next* várias vezes

- Como a lista é circular, não saberemos quando a lista inteira foi percorrida, a não ser que façamos o teste: if(aux == lista->next).

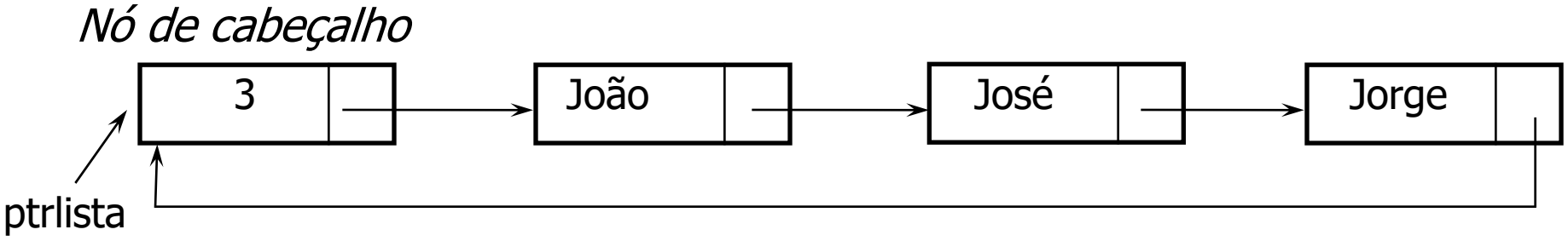
- Solução:
  - Posicionar o nó de cabeçalho como primeiro elemento da lista circular
  - Campo *info* do nó de cabeçalho deverá ser inválido para o contexto do problema, ou poderá conter um sinal que o marque como nó cabeçalho

Nó de cabeçalho



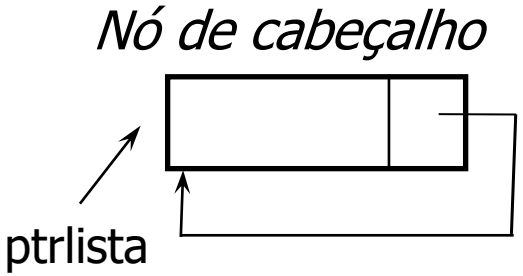
Nó de cabeçalho

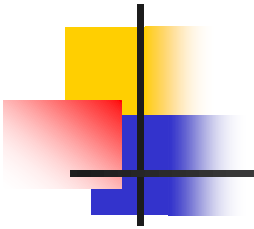
- Também conhecido como Header, Sentinela
- Possibilidade de uso:
  - Informação Adicional sobre a lista que possa ser útil na aplicação
    - Armazenar número de elementos da lista, para que não seja necessário percorrê-la contando seus elementos



Nó de cabeçalho

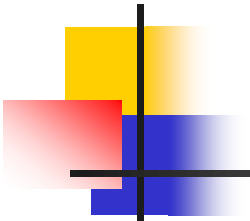
- Como representar a lista vazia?





Exercício

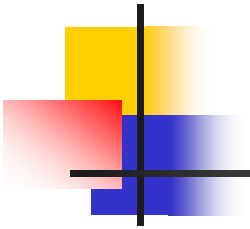
1) Considerando que cada elemento de uma lista circular é formado por um processo do sistema operacional Windows (nome e número do processo) defina um lista simplesmente encadeada circular assumindo que a mesma será implementada usando a representação com nó cabeçalho. O nó cabeçalho deverá conter o número total de processos que estão sendo executados. Em seguida desenvolva uma aplicação que permita inserir e exibir os dados de cada processo na lista.



Exercício

2) Crie um arquivo ListaSimplesmentEncadeadaCircular.c e implementa as seguintes informações de uma lista circular:
   - ✓ Definição
   - ✓ Operações
     - init
     - getnode
     - freenode
     - insere_inicio
     - listar
     - remove_inicio

→ Teste seu programa criando um menu com as opções de inserir, remover e exibir.



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

- 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. 287 (Nós de Cabeçalho) até pág. 291