

SIN 211 - Algoritmos e Estruturas de Dados

(Lista Encadeada Circular)

Profº: Joelson Antônio dos Santos

Universidade Federal de Viçosa
Instituto de Ciências Exatas e Tecnológicas
Campus de Rio Paranaíba - MG

joelsonn.santos@gmail.com
Sala: BBT 233

12 de abril de 2018

Aula de Hoje

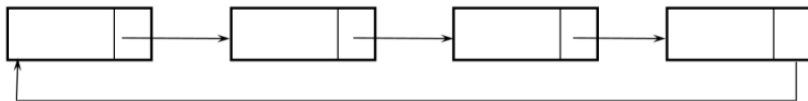
1 Lista Encadeada Circular

Créditos

O material desta aula é composto por adaptações e extensões dos originais gentilmente cedidos pelos professores **Moacir Pereira Ponti** e **Rachel Reis**.

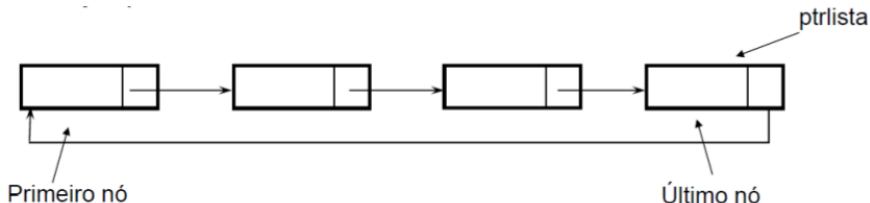
Lista Circular

- Uma lista encadeada circular é um tipo de lista onde o último elemento tem como “próximo” elemento o primeiro elemento da lista.
- Forma um ciclo.
- Qual é o primeiro nó verdadeiro? E o último nó?



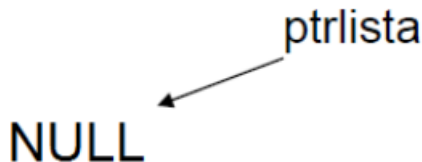
Lista Circular

- É necessário adotarmos uma convenção para determinar qual é o primeiro e o último nó.
- A convenção mais útil é permitir que o ponteiro externo aponte para o último nó da lista circular, e que o nó seguinte (por ex. **prox**) se torne o primeiro nó.



Lista Circular

- Podemos adotar que quando nosso ponteiro externo é nulo (**NULL**), nossa lista é vazia.



Lista Circular

- Adotando as mesmas implementações das aulas anteriores, tem-se que:

`CELULA*` ptrlista; // ponteiro externo para lista circular.

ptrlista->prox // referência para o primeiro nó.

- A vantagem desta implementação é podermos incluir ou remover elementos a partir do início ou final da lista.

Lista Circular - Exemplo

- Diversos processos necessitam de um mesmo recurso para funcionar.
- Os recursos só podem ser alocados para um único processo por vez.
- Depois que um processo é ativado, o próximo processo recebe os recursos.
- Após o último processo ser ativado, os recursos voltam para o primeiro processo.

Lista Circular

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

```
typedef struct sPessoa{  
    char nome[50];  
    int matricula;  
}PESSOA;  
  
typedef struct sCell{  
    PESSOA info;  
    struct sCell * prox;  
}CELULA;
```

Lista Circular - Implementação

- Inicializando a lista

```
void inicializarLista(CELULA** lista){  
    (*lista) = NULL;  
}
```

- Comparando a inicialização da lista circular com a implementação da lista encadeada simples, o que mudou?

Lista Circular - Inserir no Final


```
int inserirFim(CELULA** lista, PESSOA elemento){
    CELULA* p = criarCelula();
    if(p == NULL){
        printf("\n Erro de  alocao de memoria");
        return 0;
    }
    p->info = elemento;
    if(listaVazia(lista)){
        p->prox = p;
        (*lista) = p;
        return 1;
    }
}
```

Lista Circular - Inserir no Final (Cont.)

```
p->prox = (*lista)->prox;  
(*lista)->prox = p;  
(*lista) = p;  
return 1;
```

```
}
```

Lista Circular - Imprimir Elementos



```
void imprimirLista(CELULA** lista){
    if(listaVazia(lista)){
        printf("\n Lista Vazia!");
    }else{
        CELULA* aux = (*lista)->prox;
        printf("\n Lista: \n");
        do{
            imprimirElementos(aux->info);
            aux = aux->prox;
        }while(aux != (*lista)->prox);
    }
}
```

Lista Circular - Remover Início

```
PESSOA removerInicio(CELULA** lista){
    CELULA* removida;
    PESSOA elementoRemovido;

    // elemento vazio
    strcpy(elementoRemovido.nome, "  ");
    elementoRemovido.matricula = -1;

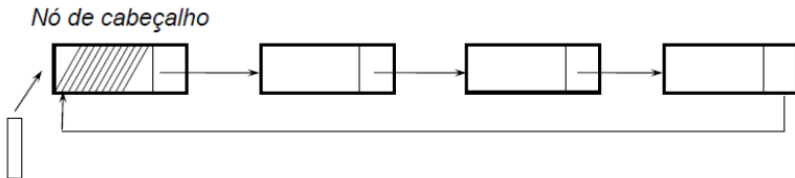
    if(listaVazia(lista)){
        printf("\n Lista vazia\n");
        return elementoRemovido;
    }
    removida = (*lista)->prox;
    elementoRemovido = removida->info;
```

Lista Circular - Remover Início - (Cont.)

```
// Se lista contém apenas um elemento
if((*lista)->prox == (*lista)){
    inicializarLista(lista);
}else{
    (*lista)->prox = removida->prox;
    // ou (*lista)->prox->prox
}
free(removida);
return elementoRemovido;
```

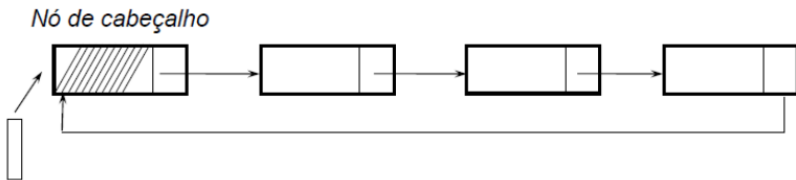
Lista Circular - Nó de Cabeçalho

- Outro tipo de implementação pode ser adotado para a lista circular.
- Esse método se baseia em criar um nó de cabeçalho para sabermos onde é o início da nossa lista.



Lista Circular - Nó de Cabeçalho

- O nó de cabeçalho será o primeiro elemento da lista.
- O campo info do nó de cabeçalho deverá ser INVÁLIDO para o contexto do problema.



Bibliografia Básica

- 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