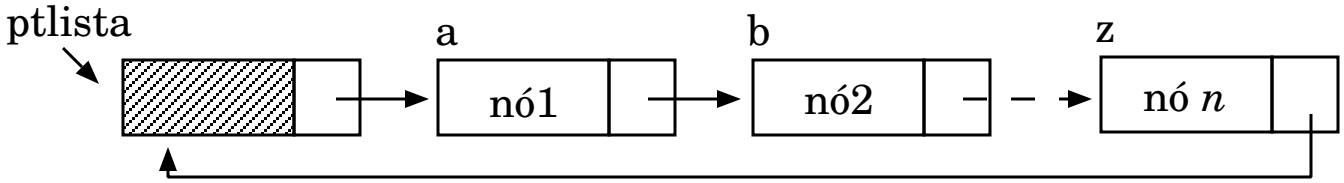
Aula 15: Listas circulares e listas duplamente encadeadas

- Modelos
- Algoritmos de busca
- Complexidade de algoritmos

Modelo de listas circulares

- Numa lista circular encadeada, obrigamos o último nó da lista a apontar para o nó cabeça



- Vantagem: eliminar a necessidade de testes de fim de lista nos algoritmos de busca

Descrição do algoritmo de busca em listas circulares encadeadas

- Coloca-se a chave x a ser procurada no nó cabeça, de forma que haja sempre resposta positiva na busca

- No fim da busca, ant e pont apontam para o penúltimo e o último nós pesquisados, respectivamente

Descrição do algoritmo de busca em listas circulares encadeadas

- Algoritmo: Busca numa lista encadeada ordenada

```
procedimento busca-cir( x, ant, pont )
    ant := ptlista
    ptlista↑.chave := x
    pont := ptlista↑.prox
    enquanto pont↑.chave < x faça
        ant := pont
        pont := pont↑.prox
    se pont ≠ ptlista e pont↑.chave = x
        então "chave localizada"
        senão "chave não localizada"
```

Complexidade do algoritmo de busca em listas circulares encadeadas

- No pior dos casos, todos os nós da lista serão percorridos. Por exemplo, basta que a chave x não seja encontrada na lista

- Assim, a complexidade é O(n), onde n é o número de nós da lista

- Exercício: Descrever um algoritmo de inserção numa lista circular encadeada
  - Sugestão: utilizar o algoritmo de busca

Tempo: 10 minutos

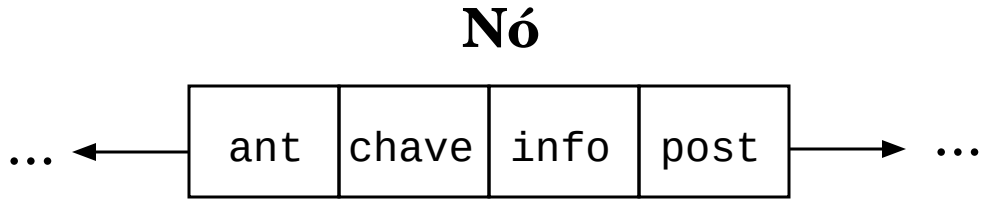
Solução do exercício

Algoritmo: Inserção de um nó em lista circular encadeada após o nó apontado por ant

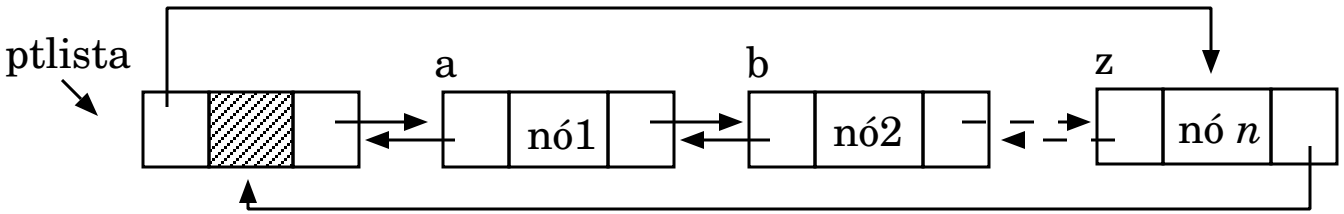
```
busca-cir( x, ant, pont )
se pont ≠ ptlista e pont↑.chave = x
    então "elemento já existe na lista"
    senão
        ocupar( pt )    % solicitar novo nó à LED
        pt↑.info  := novo_valor      }
        pt↑.chave := x               } inicializar nó
        pt↑.prox  := ant↑.prox       }
        ant↑.prox := pt              % acertar lista
```

Modelo de listas duplamente encadeadas

- Cada nó agora apresenta dois campos de ponteiros: ant (que aponta para o nó anterior na lista) e post (que aponta para o nó posterior na lista



- Vantagem: possibilidade de fazer o percurso da lista nos dois sentidos indiferentemente



Descrição do algoritmo de busca em listas duplamente encadeadas

- O algoritmo retorna indicando o nó processado ou, se este não foi encontrado, o nó que seria seu consecutivo

- Algoritmo: Busca em uma lista duplamente encadeada ordenada

```
função busca-dup( x )
    % último aponta para o final da lista
    ultimo := ptlista↑.ant
    se x ≤ ultimo↑.chave
        então
            pont := ptlista↑.post
            enquanto pont↑.chave < x faça
                pont := pont↑.post
            busca-dup := pont
        senão
            busca-dup := ptlista
```

Complexidade do algoritmo de busca em listas duplamente encadeadas

- No pior caso, é necessário percorrer todos os nós da lista. Portanto, a complexidade é O(n), onde n é o número de nós da lista

Exercícios finais

- Descreva um algoritmo de remoção em uma lista circular encadeada

- Uma palavra é um palíndromo se a seqüência de letras que a forma é a mesma, quer seja lida da esquerda para direita ou da direita para esquerda (exemplo: arara)

Exercícios finais

- Escrever um algoritmo para reconhecer se uma dada palavra é um palíndromo. Suponha que a palavra esteja representada como uma lista duplamente encadeada

- Exemplo: