Algoritmos e Estruturas de dados

Listas Circulares

Prof. Nilton Luiz Queiroz Jr.

Listas

- Imagine que se tem um ponteiro para um elemento em uma lista
  - Como faríamos para alcançar elementos posteriores a lista nesse ponteiro;
    - Não temos acesso a itens anteriores;
    - A única maneira de fazer isso seria obtendo o início da lista novamente;
- Listas circulares contornam essa situação;
  - Dado uma célula p, podemos percorrer a lista até chegar a ponteiros anteriores a p;

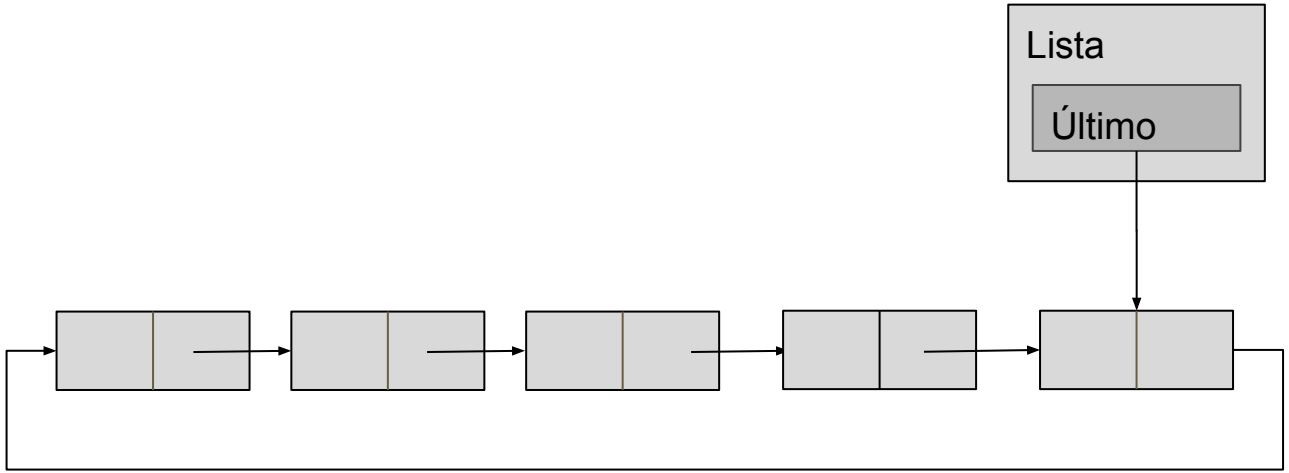
Listas Circulares

- Listas circulares são listas cujo o ponteiro de próximo do último elemento aponta para o primeiro elemento;
- Diferente das listas **não** circulares, as lisas circulares não possuem nem “primeira” nem “última” célula “natural”;
  - As listas encadeadas **não** circulares tem um primeira e uma última célula natural;
    - A primeira célula é apontada pela estrutura lista;
      - Ou pela sentinela;
    - A última célula é aterrado;

Lista Circular

- A implementação de uma lista circular tem algumas diferenças com de listas não circulares;
  - A estrutura de dados lista não precisa de um ponteiro para o início e outro para o fim;
    - Apenas um ponteiro para o fim, e o sucessor do fim é o início;

Implementação de uma Lista Circular



Implementação de uma Lista Circular

- Os agregados heterogêneos para uma lista são:

```
struct tipo_item{
    int chave;
    /*outros campos*/
};

struct tipo_celula{
    struct tipo_item item;
    struct tipo_celula *prox;
};

struct tipo_lista{
    struct tipo_celula *ultimo;
};
```

Implementação de uma Lista Circular

- A inicialização de uma lista circular pode ser feita aterrando o ponteiro do último item;

```
void inicializa(struct tipo_lista *l){
    l->ultimo=NULL;
}
```

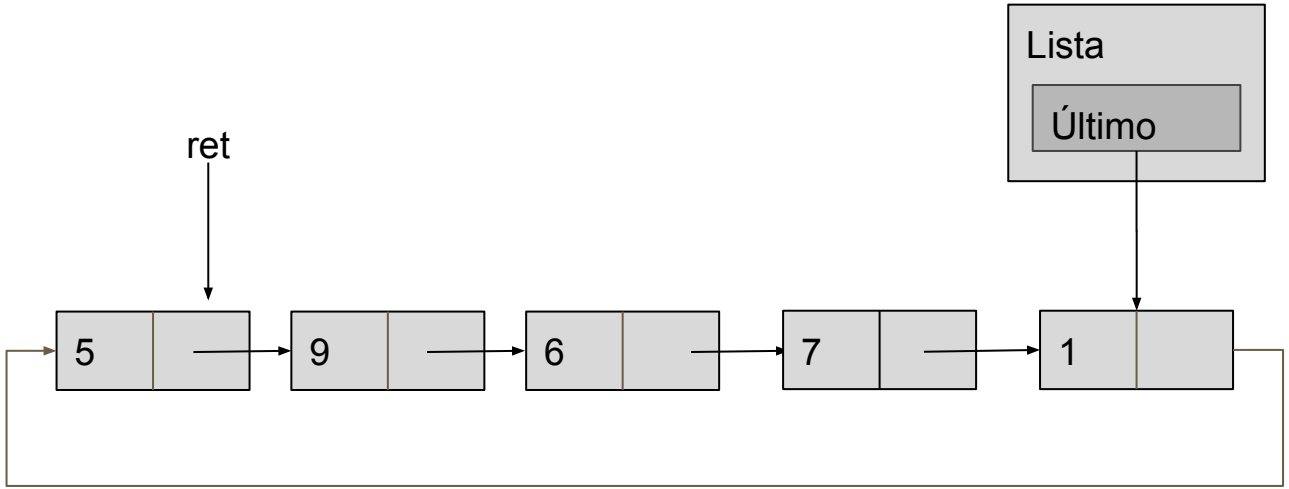
Implementação de uma Lista Circular

- Para verificar se a lista está vazia basta verificar se o ponteiro que aponta para o final da lista está aterrado;

```
int vazia(struct tipo_lista *l){
    return l->ultimo == NULL;
}
```

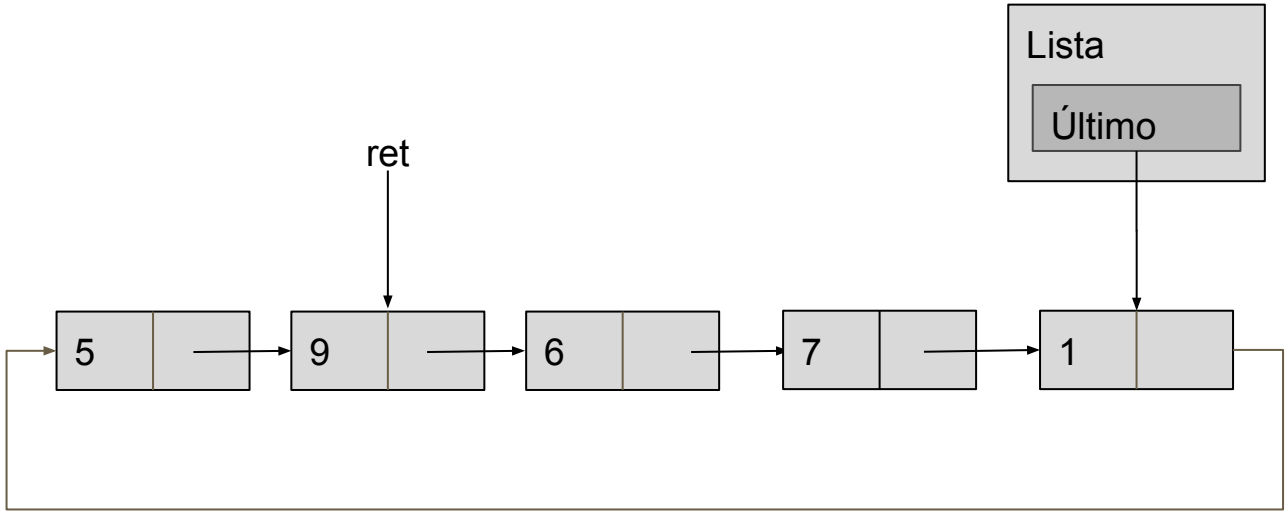
Implementação de uma Lista Circular

- Busca pelo valor 7



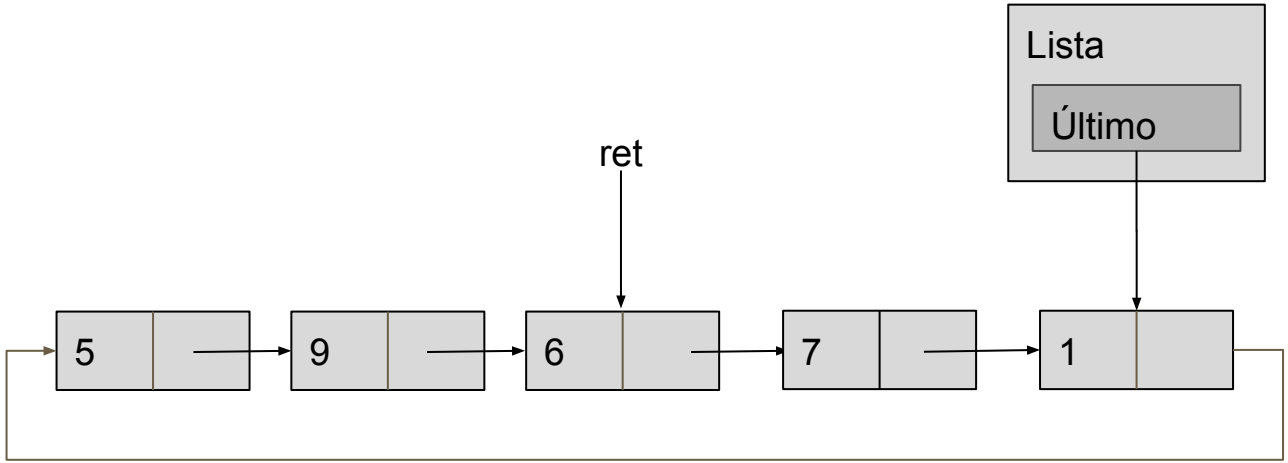
Implementação de uma Lista Circular

- Busca pelo valor 7



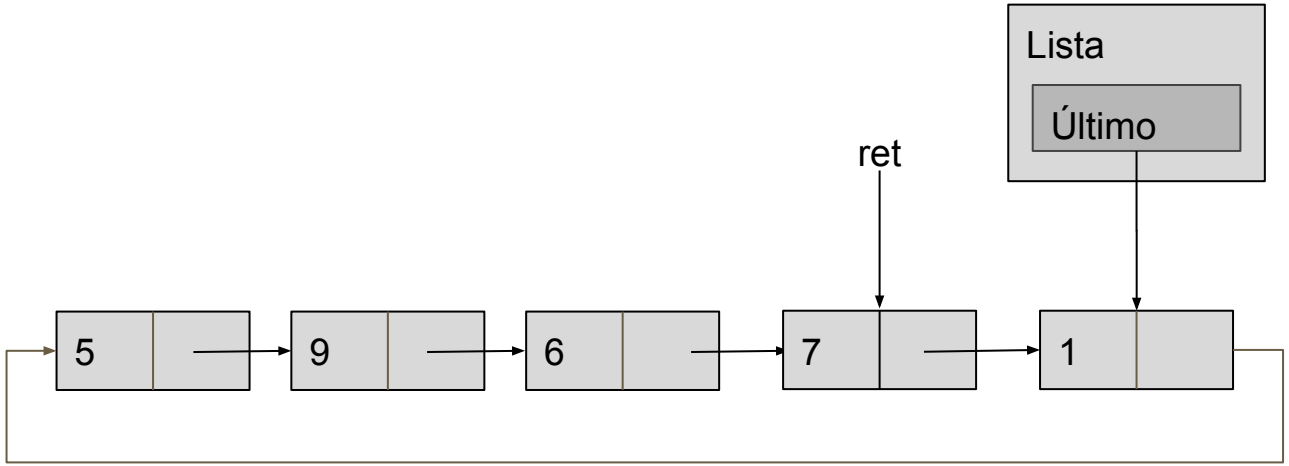
Implementação de uma Lista Circular

- Busca pelo valor 7



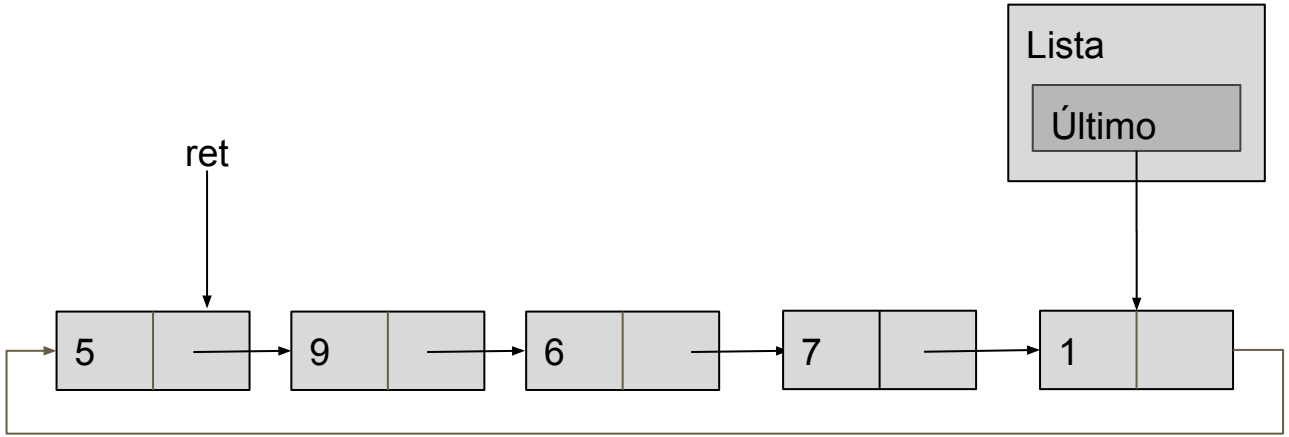
Implementação de uma Lista Circular

- Busca pelo valor 7
  - Valor encontrado, então sai do laço
    - Retorna-se o local que o ponteiro está apontando;



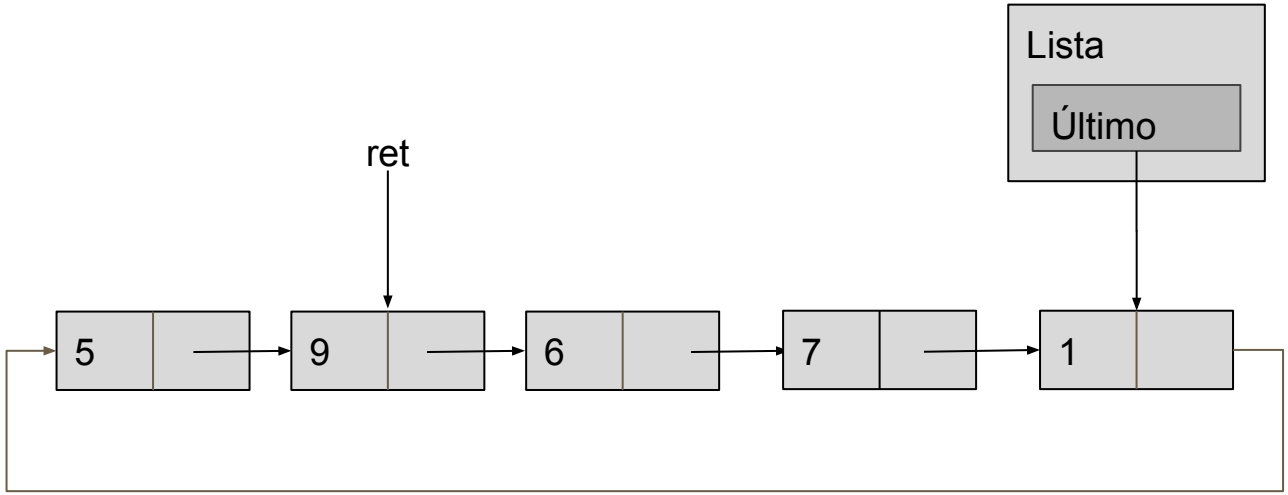
Implementação de uma Lista Circular

- Busca pelo valor 15



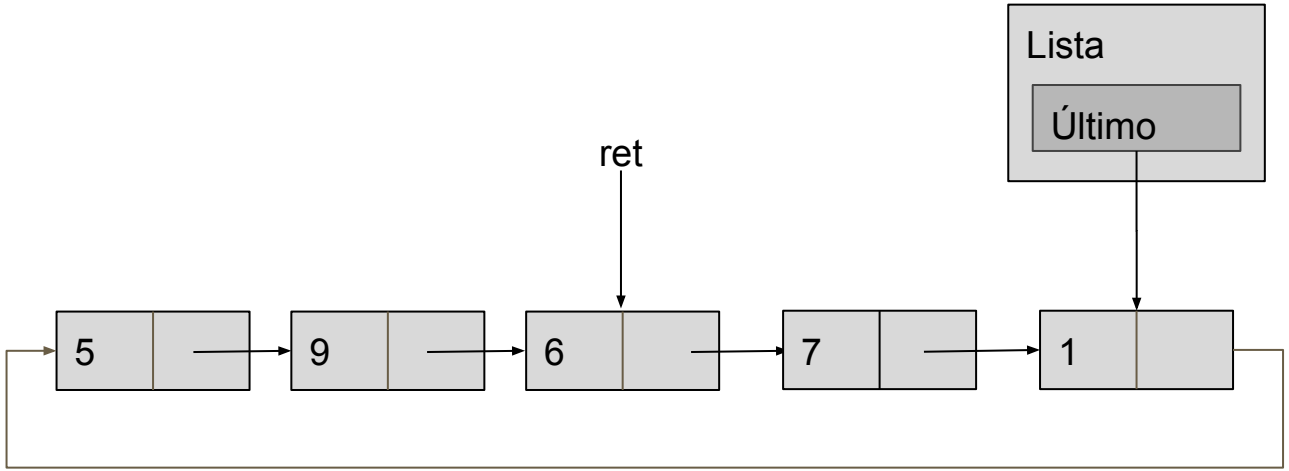
Implementação de uma Lista Circular

- Busca pelo valor 15



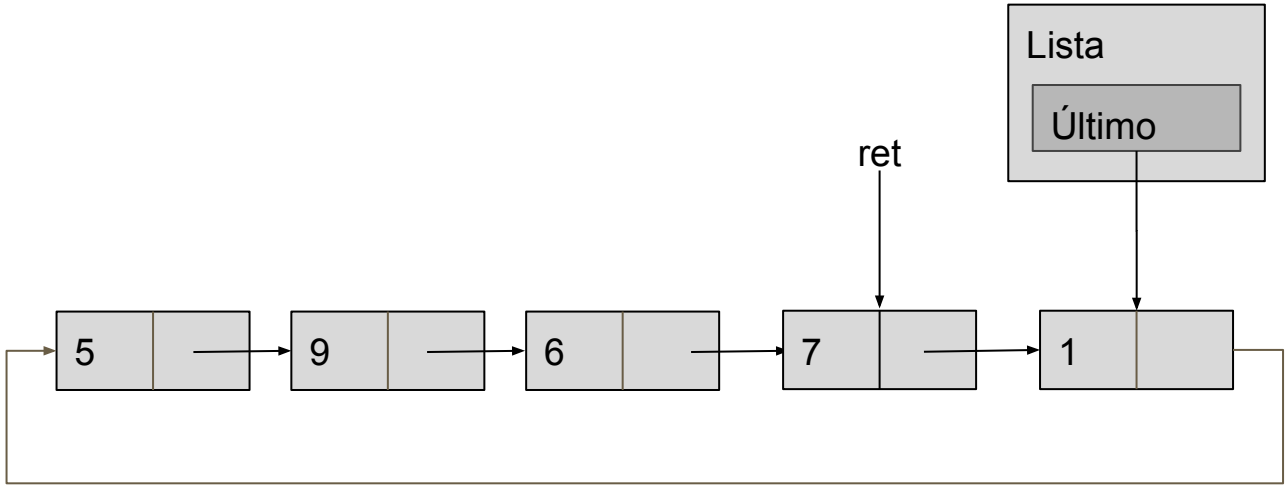
Implementação de uma Lista Circular

- Busca pelo valor 15



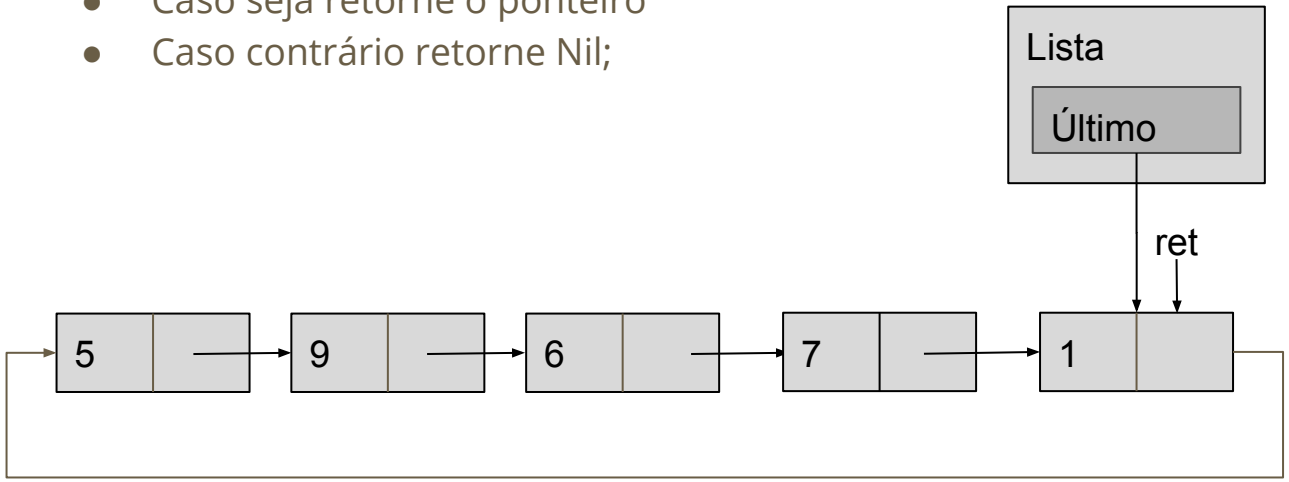
Implementação de uma Lista Circular

- Busca pelo valor 15



Implementação de uma Lista Circular

- Busca pelo valor 15
  - Chegou no último elemento;
    - Sai do laço e pergunta se os valor é o valor procurado;
      - Caso seja retorne o ponteiro
      - Caso contrário retorne Nil;



Implementação de uma Lista Circular

- Busca:
  - Deve-se verificar o caso de lista vazia;

```
struct tipo_celula *busca(struct tipo_lista *l, int chave){
    struct tipo_celula *ret,*ult;
    if(vazia(l)){
        return NULL;
    }else{
        ret = l->ultimo->prox;
        ult = l->ultimo;
        while((ret!=ult) && (chave!=ret->item.chave)){
            ret=ret->prox;
        }
        if(chave==ret->item.chave){
            return ret;
        }else{
            return NULL;
        }
    }
}
```

Implementação de uma Lista Circular

- Inserção no fim:
  - Fica mais fácil quando se usa um ponteiro para o último elemento;
  - Feita de maneira similar a inserção de uma lista linear (não circular);
    - A principal diferença é que o último elemento deverá apontar para o primeiro
  - Deve-se tratar o caso de lista vazia;

Implementação de uma Lista Circular

- Inserção no fim

```
void insere_ultimo(struct tipo_lista *l, struct tipo_item x){
    /*algoritmo*/
}
```

Implementação de uma Lista Circular

- Inserção no fim

```
void insere_ultimo(struct tipo_lista *l, struct tipo_item x){
    struct tipo_celula *novo;
    novo=(struct tipo_celula *)malloc(sizeof(struct tipo_celula));
    novo->item=x;
    if(vazia(l)){
        novo->prox=novo;
    }else{
        novo->prox=l->ultimo->prox;
        l->ultimo->prox=novo;
    }
    l->ultimo=novo;
}
```

Implementação de uma Lista Circular

- Inserção no início;
  - É importante tratar o caso de lista vazia;

Implementação de uma Lista Circular

- Inserção no início:

```
void insere_primeiro(struct tipo_lista *l, struct tipo_item x){
    /*algoritmo*/
}
```

Implementação de uma Lista Circular

- Inserção no início:

```
void insere_primeiro(struct tipo_lista *l, struct tipo_item x){
    struct tipo_celula *novo;
    if(vazia(l)){
        insere_ultimo(l,x);
    }else{
        novo=(struct tipo_celula *)malloc(sizeof(struct tipo_celula));
        novo->item=x;
        novo->prox=l->ultimo->prox;
        l->ultimo->prox=novo;
    }
}
```

Implementação de uma Lista Circular

- Remoção no início;
  - Não remover em lista vazia;
  - Tratar remoção do último nó;
    - Quando a lista for ficar vazia;

Implementação de uma Lista Circular

- Remoção no início

```
int remove_primeiro(struct tipo_lista *l, struct tipo_item *x){
    /*algoritmo*/
}
```

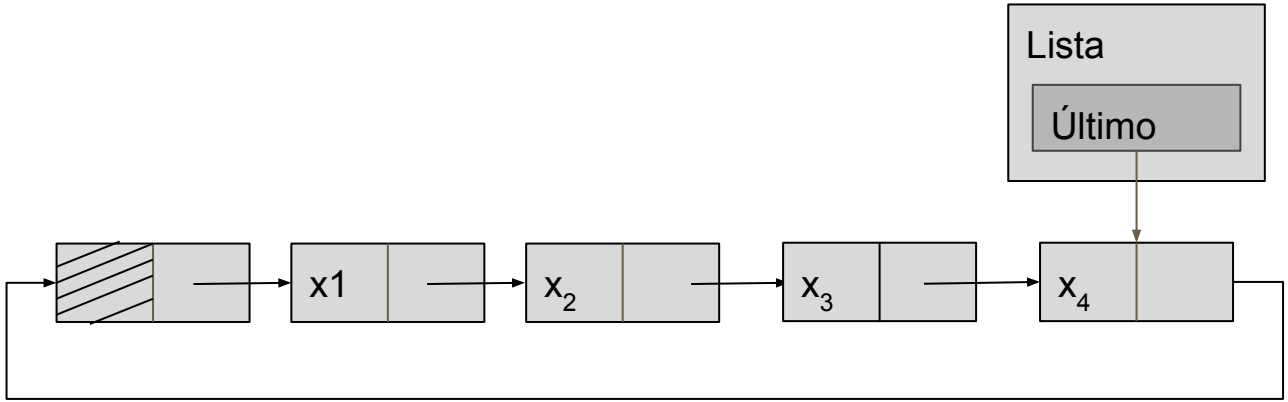
Implementação de uma Lista Circular

- Remoção no início

```
int remove_primeiro(struct tipo_lista *l, struct tipo_item *x){
    struct tipo_celula *ptr;
    if(!vazia(l)){
        ptr=l->ultimo->prox;
        *x=ptr->item;
        if(l->ultimo->prox == l->ultimo){
            l->ultimo=NULL;
        }else{
            l->ultimo->prox = ptr->prox;
        }
        free(ptr);
    }else{
        return 0;
    }
}
```

Listas Circulares

- Listas circulares também podem ter ou não sentinela;
  - A lista aqui implementada não tem sentinela;
- A sentinela nas listas circulares é colocada antes do primeiro elemento e depois do último
  - O último aponta para ela, e o primeiro é apontado por ela;



Exercícios

1. Implemente a remoção por chave para lista circular.
2. Implemente a remoção no fim de uma lista circular.
3. Implemente a inserção após a i-ésima posição para lista circular.
4. Implemente as operações de inserção em uma lista circular que não permita chaves repetidas;
5. O problema de Josephus é descrito a seguir:
   Há um grupo de soldados circundado por uma força inimiga esmagadora. Não há esperanças de vitória sem a chegada de reforços, mas existe apenas um cavalo disponível para escapar. Os soldados entram num acordo para determinar qual deles deverá escapar e trazer ajuda. Para resolver tal situação, os soldados fazem o seguinte

Exercícios

- Eles formam um círculo e um número n é sorteado num chapéu.
- Um de seus nomes é sorteado, e começando pelo nome sorteado, eles começam a contar ao longo do círculo em sentido horário.
- Quando a contagem alcança n, esse soldado é retirado e a contagem reinicia com o soldado seguinte;
- O último soldado que restar deverá montar no cavalo e escapar.

Considerando N e uma lista de Nomes, diga a ordem de eliminação dos soldados, e qual será o soldado que irá escapar;
Assuma que o primeiro nome digitado foi o nome sorteado;

Referências

Tenenbaum, A. A.; Langsam, Y.; Augenstein, M. J. **Estruturas de dados usando C**. São Paulo: MAKRON Books do Brasil Editora Ltda, 1995.