

Listas Circulares

Listas lineares encadeadas possuem alguma deficiência?

Sim.

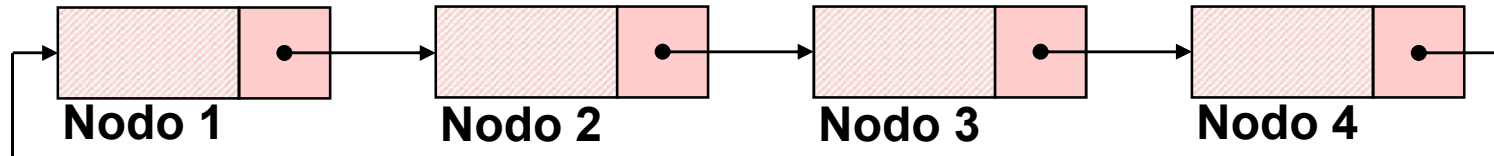
Qual?

Dado um ponteiro p para um nodo de uma lista linear encadeada, não podemos atingir nenhum nodo que antecede o apontado por p .

Contudo, se fizermos uma pequena alteração na estrutura de lista que temos trabalhado, fazendo com que o campo *next* do último nodo ao invés de conter **NULL** armazene o endereço do primeiro nodo da lista, resolveremos este problema.

Listas Circulares

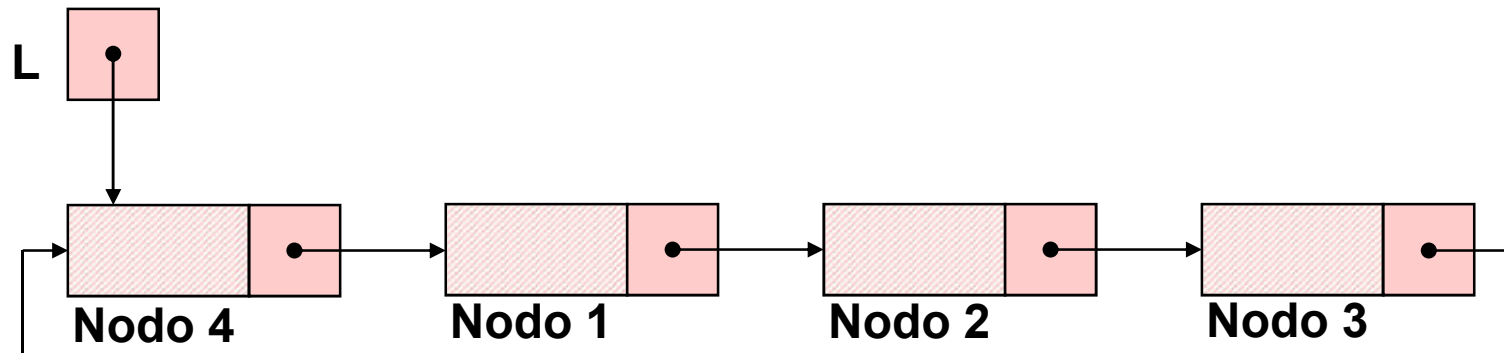
Este tipo de lista é denominado lista circular, possuindo a forma abaixo:



Uma pergunta surge: para qual elemento apontar para que se tenha uma referência a lista circular?

Listas Circulares

Uma convenção útil é fazer com que o ponteiro externo para a lista circular aponte para o último elemento.



Pois, desta forma, se tem acesso direto ao último e primeiro elemento.

Listas Circulares

Como vocês poderão observar a definição do TAD LISTA_CIRCULAR é praticamente a mesma do TAD LISTA_ENC, apenas algumas pequenas alterações são necessárias nas implementações das operações.

```
typedef struct nodo  
{  
    int inf;  
    struct nodo * next;  
}NODO;  
typedef NODO * LISTA_CIRCULAR;  
void cria_lista (LISTA_CIRCULAR *);  
int eh_vazia (LISTA_CIRCULAR);  
int tam (LISTA_CIRCULAR);  
void ins (LISTA_CIRCULAR *, int, int);  
int recup (LISTA_CIRCULAR, int);  
void ret (LISTA_CIRCULAR *, int);
```

Listas Circulares

Com base no que foi visto implemente a operação `cria_lista()` que compõem o TAD `LISTA_CIRCULAR`.

```
typedef struct nodo
{
    int inf;
    struct nodo * next;
}NODO;
typedef NODO * LISTA_CIRCULAR;
void cria_lista (LISTA_CIRCULAR *);
int eh_vazia (LISTA_CIRCULAR);
int tam (LISTA_CIRCULAR);
void ins (LISTA_CIRCULAR *, int, int);
int recup (LISTA_CIRCULAR, int);
void ret (LISTA_CIRCULAR *, int);
```

```
void cria_lista (LISTA_CIRCULAR *pl)  
{  
    *pl=NULL;  
}
```

Listas Circulares

Com base no que foi visto implemente a operação `eh_vazia()` que compõem o TAD `LISTA_CIRCULAR`.

```
typedef struct nodo
{
    int inf;
    struct nodo * next;
}NODO;
typedef NODO * LISTA_CIRCULAR;
void cria_lista (LISTA_CIRCULAR *);
int eh_vazia (LISTA_CIRCULAR);
int tam (LISTA_CIRCULAR);
void ins (LISTA_CIRCULAR *, int, int);
int recup (LISTA_CIRCULAR, int);
void ret (LISTA_CIRCULAR *, int);
```



```
int eh_vazia (LISTA_CIRCULAR l)  
{  
    return (l == NULL);  
}
```

Listas Circulares

Com base no que foi visto implemente a operação tam() que compõem o TAD LISTA_CIRCULAR.

```
typedef struct nodo
{
    int inf;
    struct nodo * next;
}NODO;
typedef NODO * LISTA_CIRCULAR;
void cria_lista (LISTA_CIRCULAR *);
int eh_vazia (LISTA_CIRCULAR);
int tam (LISTA_CIRCULAR);
void ins (LISTA_CIRCULAR *, int, int);
int recup (LISTA_CIRCULAR, int);
void ret (LISTA_CIRCULAR *, int);
```

```
int tam (LISTA_CIRCULAR l)
{
    if (l==NULL)
        return (0);
    else
    {
        LISTA_CIRCULAR aux; /* NODO *aux; */
        int cont;
        for (cont=1, aux=l->next; aux!=l ; cont++)
            aux = aux->next;
        return (cont);
    }
}
```

```
}
```

Listas Circulares

Com base no que foi visto implemente a operação `ins()` que compõem o TAD `LISTA_CIRCULAR`.

```
typedef struct nodo
{
    int inf;
    struct nodo * next;
}NODO;
typedef NODO * LISTA_CIRCULAR;
void cria_lista (LISTA_CIRCULAR *);
int eh_vazia (LISTA_CIRCULAR);
int tam (LISTA_CIRCULAR);
void ins (LISTA_CIRCULAR *, int, int);
int recup (LISTA_CIRCULAR, int);
void ret (LISTA_CIRCULAR *, int);
```

```

void ins (LISTA_CIRCULAR *pl, int v, int k)
{
    NODO *novo;
    if (k < 1 || k > tam(*pl)+1)
    {
        printf ("\nERRO! Posição invalida para insercao.\n");
        exit (1);
    }
    novo = (NODO *) malloc (sizeof(NODO));
    if (!novo)
    {
        printf ("\nERRO! Memoria insuficiente!\n");
        exit (2);
    }
}

```

```
novo->inf = v;
if (*pl==NULL) {
    novo->next=novo;
    *pl = novo; }
else
{
    LISTA_CIRCULAR aux=*pl;
    if (k==tam(*pl)+1)
        *pl=novo;
    for (; k>1; aux=aux->next, k--);
    novo->next = aux->next;
    aux->next = novo;    } }
```

Listas Circulares

Com base no que foi visto implemente a operação `recup()` que compõem o TAD `LISTA_CIRCULAR`.

```
typedef struct nodo
{
    int inf;
    struct nodo * next;
}NODO;
typedef NODO * LISTA_CIRCULAR;
void cria_lista (LISTA_CIRCULAR *);
int eh_vazia (LISTA_CIRCULAR);
int tam (LISTA_CIRCULAR);
void ins (LISTA_CIRCULAR *, int, int);
int recup (LISTA_CIRCULAR, int);
void ret (LISTA_CIRCULAR *, int);
```