

Alocação Encadeada

Com base em nossos novos conhecimentos adquiridos, podemos agora definir um novo TAD `LISTA_ENC_ORD`, no qual os elementos encontram-se ordenados de forma crescente ou decrescente, ou seja, no caso da ordenação crescente, o primeiro elemento é menor que o segundo, que por sua vez é menor que o terceiro e assim sucessivamente. (Para exemplificar, consideraremos a ordem crescente)

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

Alocação Encadeada

Utilizando como base a implementação do TAD LISTA_ENC faça as devidas adequações para implementar o TAD LISTA_ENC_ORD.

Com uma pequena análise, percebe-se que a única operação que requer alteração no TAD LISTA_ENC para o TAD LISTA_ENC_ORD é a operação de inserção.

Veremos agora a implementação desta operação.

Alocação Encadeada - Exercício

Implemente, no TAD LISTA_ENC_ORD, a seguinte operação:

```
int ret_com_base_no_valor (LISTA_ENC_ORD *, int);
```

a qual recebe uma referência para uma lista e um valor que deve ser retirado desta. Caso o valor pertença à lista e conseqüentemente seja retirado a operação retorna 1; caso contrário retorna 0.

Alocação Encadeada – Nós de cabeçalho

Ocasionalmente, é desejável manter um nó adicional no início de uma lista.

Esse nó não representa um item (elemento) na lista e é chamado *nó de cabeçalho* ou *cabeçalho de lista*.

A parte *inf* deste nó, frequentemente, é usada para manter informações globais sobre a lista.

Alocação Encadeada – Nós de cabeçalho

Por exemplo, a parte *inf* do nó de cabeçalho pode ser usada para armazenar o número de elementos na lista. No caso particular em que temos uma lista com o campo *inf* numérico, um nó de lista pode ser utilizado como nó cabeçalho.

Definiremos agora, um TAD LISTA_ENC_NC, o qual representa uma lista linear encadeada dinamicamente com a presença de um nó de cabeçalho contendo no campo *inf* o número de elementos contidos na lista.

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

Alocação Encadeada

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

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



```
void cria_lista (LISTA_ENC_NC *pl)  
{  
    *pl = (NODO *) malloc (sizeof(NODO));  
    if (!*pl)  
    {  
        printf ("\nERRO! Memoria  
insuficiente!\n");  
        exit (2);  
    }  
    (*pl)->inf = 0;  
    (*pl)->next = NULL;  
}
```

Alocação Encadeada

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

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

```
int eh_vazia (LISTA_ENC_NC l)  
{  
    return (l->inf==0);  
}
```

```
int eh_vazia (LISTA_ENC_NC l)  
{  
    return (!(l->next));  
}
```

Alocação Encadeada

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

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

```
int tam (LISTA_ENC_NC l)  
{  
    return (l->inf);  
}
```

Alocação Encadeada

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

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

Alocação Encadeada

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

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


Alocação Encadeada

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

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