```
#if ! defined( LISTA_ )
   #define LISTA_
/***************************************************************************
*
*  $MCD Módulo de definição: LIS  Lista duplamente encadeada
*
*  Arquivo gerado:              LISTA.h
*  Letras identificadoras:      LIS
*
*  Nome da base de software:    Arcabouço para a automação de testes de
programas redigidos em C
*  Arquivo da base de software: D:\AUTOTEST\PROJETOS\LISTA.BSW
*
*  Projeto: Trabalho 2 - Programação Modular
*  Autores: avs - Arndt von Staa
*           GB - Gustavo Bach
*           JG - João Lucas Gardenberg
*           MV - Michelle Valente
*
*  $HA Histórico de evolução:
*     Versão  Autor      Data          Observações
*     5.00    GM,JG,MV   12/abr/2014   todas as funções retornam condições de
retorno
*     5.00    GB,JG,MV   11/abr/2014   adicionar função de alterar valor do
elemento
*     4.00    avs        01/fev/2006   criar linguagem script simbólica
*     3.00    avs        08/dez/2004   uniformização dos exemplos
*     2.00    avs        07/jul/2003   unificação de todos os módulos em um
só projeto
*     1.00    avs        16/abr/2003   início desenvolvimento
*
*  $ED Descrição do módulo
*     Implementa listas genéricas duplamente encadeadas.
*     Podem existir n listas em operação simultaneamente.
*     As listas possuem uma cabeça encapsulando o seu estado.
*
*     Cada lista é homogênea quanto ao tipo dos dados que armazena.
*     Cada elemento da lista referencia o valor que contém.
*
*     Os ponteiros para os dados são copiados para elementos da lista.
*        Não é copiado o valor apontado por estes ponteiros.
*
*     O controle da destruição do valor de um elemento a ser excluído
*        é realizado por uma função fornecida pelo usuário.
*
*     Cada lista referencia uma função que determina como devem ser
*        desalocados os dados nela contidos.
*
*     A função de liberação dos valores contidos nos elementos deve
```

```
*        assegurar a liberação de todos os espaços referênciados pelo
*        valor contido em um elemento.
*        Esta função é chamada antes de se desalocar um elemento
*        de uma lista.
*        Caso não seja necessário desalocar o valor referenciado pelo
*        elemento, o ponteiro para a função de liberação poderá ser NULL .
*        Caso o elemento da lista seja a única âncora do valor referenciado,
*        esta função deve promover a destruição (free) desse valor e
*        de todos os dados nele ancorados.
*
***************************************************************************/

#if defined( LISTA_OWN )
   #define LISTA_EXT
#else
   #define LISTA_EXT extern
#endif

/***** Declarações exportadas pelo módulo *****/

/* Tipo referência para uma lista */

typedef struct LIS_tagLista * LIS_tppLista ;


/***************************************************************************
*
*  $TC Tipo de dados: LIS Condições de retorno
*
*
*  $ED Descrição do tipo
*     Condições de retorno das funções da lista.
*
***************************************************************************/

   typedef enum {

         LIS_CondRetOK ,
               /* Concluiu corretamente */

         LIS_CondRetListaVazia ,
               /* A lista não contém elementos */

         LIS_CondRetFimLista ,
               /* Foi atingido o fim de lista */

         LIS_CondRetNaoAchou ,
               /* Não encontrou o valor procurado */
```

```
         LIS_CondRetFaltouMemoria ,
               /* Faltou memória ao tentar criar um elemento de lista */

         LIS_CondRetListaNaoExiste
               /* A lista não existe */

   } LIS_tpCondRet ;


/***************************************************************************
*
*  $FC Função: LIS  &Criar lista
*
*  $ED Descrição da função
*     Cria uma lista genérica duplamente encadeada.
*     Os possíveis tipos são desconhecidos a priori.
*     A tipagem é implicita.
*     Não existe identificador de tipo associado à lista.
*
*  $EP Parâmetros
*     $P pLista        - ponteiro para a lista a ser criada.
*                        Este ponteiro será utilizado pelas funções
*                        que manipulem esta lista.
*     $P ExcluirValor  - ponteiro para a função que processa a
*                        exclusão do valor referenciado pelo elemento
*                        a ser excluído.
*                        Ver descrição do módulo.
*
*  $FV Valor retornado
*     LIS_CondRetOK            - se executou corretamente.
*     LIS_CondRetFaltouMemoria - se faltou memória para alocar o espaço da
lista.
*
***************************************************************************/

   LIS_tpCondRet LIS_CriarLista( LIS_tppLista * pLista,
             void   ( * ExcluirValor ) ( void * pDado ) ) ;


/***************************************************************************
*
*  $FC Função: LIS  &Destruir lista
*
*  $ED Descrição da função
*     Destrói a lista fornecida.
*     O parâmetro ponteiro para a lista não é modificado.
*     Se ocorrer algum erro durante a destruição, a lista resultará
```

```
*     estruturalmente incorreta.
*     OBS. não existe previsão para possíveis falhas de execução.
*
*  $EP Parâmetros
*     $P pLista - ponteiro para a lista a ser destruída.
*
*  $FV Valor retornado
*     LIS_CondRetOK             - se destruiu sem problemas.
*     LIS_CondRetListaNaoExiste - se a lista a ser destruída não existir.
*
***************************************************************************/

   LIS_tpCondRet LIS_DestruirLista( LIS_tppLista pLista ) ;


/***************************************************************************
*
*  $FC Função: LIS  &Esvaziar lista
*
*  $ED Descrição da função
*     Elimina todos os elementos, sem contudo eliminar a lista.
*
*  $EP Parâmetros
*     pLista - ponteiro para a lista a ser esvaziada.
*
*  $FV Valor retornado
*     LIS_CondRetOK             - se esvaziou sem problemas.
*     LIS_CondRetListaNaoExiste - se a lista a ser esvaziada não existir.
*
***************************************************************************/

   LIS_tpCondRet LIS_EsvaziarLista( LIS_tppLista pLista ) ;


/***************************************************************************
*
*  $FC Função: LIS  &Inserir elemento antes
*
*  $ED Descrição da função
*     Insere novo elemento antes do elemento corrente.
*     Caso a lista esteja vazia, insere o primeiro elemento da lista.
*
*  $EP Parâmetros
*     pLista - ponteiro para a lista onde deve ser inserido o elemento.
*     pValor - ponteiro para o valor do novo elemento.
*              Pode ser NULL.
*
*  $FV Valor retornado
*     LIS_CondRetOK            - se inseriu o elemento sem problemas.
```

```
*     LIS_CondRetFaltouMemoria - se faltou memória para a alocação do
*                                elemento a ser inserido.
*
***************************************************************************/

   LIS_tpCondRet LIS_InserirElementoAntes( LIS_tppLista pLista ,
                                           void * pValor         ) ;


/***************************************************************************
*
*  $FC Função: LIS  &Inserir elemento após
*
*  $ED Descrição da função
*     Insere novo elemento após o elemento corrente.
*     Caso a lista esteja vazia, insere o primeiro elemento da lista.
*
*  $EP Parâmetros
*     Parâmetros
*        pLista - ponteiro para a lista onde deve ser inserido o elemento.
*        pValor - ponteiro para o valor do novo elemento.
*                 Pode ser NULL.
*
*
*  $FV Valor retornado
*     LIS_CondRetOK            - se inseriu o elemento sem problemas.
*     LIS_CondRetFaltouMemoria - se faltou memória para a alocação do
*                                elemento a ser inserido.
*
***************************************************************************/

   LIS_tpCondRet LIS_InserirElementoApos( LIS_tppLista pLista ,
                                          void * pValor         ) ;


/***************************************************************************
*
*  $FC Função: LIS  &Excluir elemento
*
*  $ED Descrição da função
*     Exclui o elemento corrente da lista dada.
*     Se existir o elemento à esquerda do corrente será o novo corrente.
*     Se não existir e existir o elemento à direita, este se tornará
corrente.
*     Se este também não existir a lista tornou-se vazia.
*
*  $EP Parâmetros
*     pLista    - ponteiro para a lista na qual deve excluir.
*
```

```
*  $FV Valor retornado
*     LIS_CondRetOK         - se excluiu o elemento sem problemas.
*     LIS_CondRetListaVazia - se a lista cujo elemento deveria ser excluído
*                             estiver vazia.
*
***************************************************************************/

   LIS_tpCondRet LIS_ExcluirElemento( LIS_tppLista pLista ) ;


/***************************************************************************
*
*  $FC Função: LIS  &Obter referência para o valor contido no elemento
*
*  $ED Descrição da função
*     Obtem a referência para o valor contido no elemento corrente da lista
*
*  $EP Parâmetros
*     pLista - ponteiro para a lista de onde se quer o valor
*     pValor - ponteiro que receberá a referência para o valor contido
*              no elemento.
*
*  $FV Valor retornado
*     LST_CondRetOK             - se o valor foi passado por referencia sem
problemas.
*     LST_CondRetListaNaoExiste - se a lista passada para a função não
existir.
*     LST_CondRetListaVazia     - se a lista passada para a função estiver
vazia.
*
***************************************************************************/

   LIS_tpCondRet LIS_ObterValor( LIS_tppLista pLista,
                                 void ** pValor         ) ;


/***************************************************************************
*
*  $FC Função: LIS  &Ir para o elemento inicial
*
*  $ED Descrição da função
*     Torna corrente o primeiro elemento da lista.
*
*  $EP Parâmetros
*     pLista - ponteiro para a lista a manipular.
*
*  $FV Valor retornado
*     LST_CondRetOK             - se foi para o início sem problemas.
```

```
*     LST_CondRetListaNaoExiste - se a lista passada para a função não
existir.
*       LST_CondRetListaVazia     - se a lista passada para a função estiver
vazia.
*
*****************************************************************/

   LIS_tpCondRet LIS_IrInicioLista( LIS_tppLista pLista ) ;


/*****************************************************************
*
*  $FC Função: LIS  &Ir para o elemento final
*
*  $ED Descrição da função
*     Torna corrente o elemento final da lista.
*
*  $EP Parâmetros
*     pLista - ponteiro para a lista a manipular
*
*  $FV Valor retornado
*     LST_CondRetOK             - se foi para o final sem problemas.
*     LST_CondRetListaNaoExiste - se a lista passada para a função não
existir.
*       LST_CondRetListaVazia     - se a lista passada para a função estiver
vazia.
*
*****************************************************************/

   LIS_tpCondRet LIS_IrFinalLista( LIS_tppLista pLista ) ;


/*****************************************************************
*
*  $FC Função: LIS  &Avançar elemento
*
*  $ED Descrição da função
*     Avança o elemento corrente numElem elementos na lista
*     Se numElem for positivo avança em direção ao final
*     Se numElem for negativo avança em direção ao início
*     numElem pode ser maior do que o númro de elementos existentes na
*               direção desejada
*     Se numElem for zero somente verifica se a lista está vazia
*
*  $EP Parâmetros
*     pLista  - ponteiro para a lista a ser manipulada.
*     numElem - o número de elementos a andar.
*
*  $FV Valor retornado
```

```
*     CondRetOK         - se numElem elementos tiverem sido andados.
*     CondRetFimLista   - se encontrou o fim da lista antes de andar numElem
*                         elementos.
*     CondRetListaVazia - se a lista passada para a função estiver vazia.
*
***************************************************************************/

   LIS_tpCondRet LIS_AvancarElementoCorrente( LIS_tppLista pLista ,
                                              int numElem           ) ;


/***************************************************************************
*
*  $FC Função: LIS  &Procurar elemento contendo valor
*
*  $ED Descrição da função
*     Procura o elemento que referencia o valor dado.
*     A função compara ponteiro e não conteúdo apontado.
*
*  $EP Parâmetros
*     pLista  - ponteiro para a lista onde procura
*     pValor  - ponteiro para o valor procurado
*               Pode ser NULL
*
*  $FV Valor retornado
*     LIS_CondRetOK           - se encontrou.
*                               O elemento corrente é o primeiro elemento do
*                               elemento corrente inclusive para o fim da
lista
*                               e que contém o ponteiro procurado
*     LIS_CondRetNaoEncontrou - se o ponteiro não foi encontrado
*                               O elemento corrente continua o mesmo
*     LIS_CondRetListaVazia   - se a lista passada para a função estiver
vazia.
*
***************************************************************************/

   LIS_tpCondRet LIS_ProcurarValor( LIS_tppLista pLista ,
                                    void * pValor         ) ;


/***************************************************************************
*
*  $FC Função: LIS  &Alterar valor de um elemento
*
*  $ED Descrição da função
*     Altera o valor do elemento corrente.
*
*  $EP Parâmetros
```

```
*     pLista  - ponteiro para a lista cujo elemento será alterado.
*     pValor  - ponteiro para o Valor a ser inserido no lugar do
*               valor anterior.
*
*  $FV Valor retornado
*     LIS_CondRetOK               - se o valor for alterado sem problemas.
*     LIS_CondRetListaNaoExiste   - se a lista passada para a função não
existir.
*     LIS_CondRetListaVazia       - se a lista passada para a função estiver
vazia.
*
***************************************************************************/

   LIS_tpCondRet LIS_AlterarValor( LIS_tppLista pLista,
                                   void * pValor          ) ;


#undef LISTA_EXT

/********** Fim do módulo de definição: LIS  Lista duplamente encadeada
**********/

#else
#endif
```