```
/***************************************************************************
*  $MCI Módulo de implementação: Módulo Valor
*
*  Arquivo gerado:            VALOR.C
*  Letras identificadoras:    VAL
*
*  Nome da base de software:  Exemplo de teste automatizado
*
*  Projeto: Trabalho 2 - Programação Modular
*  Autores: GB - Gustavo Bach
*           JG - João Lucas Gardenberg
*           MV - Michelle Valente
*
*  $HA Histórico de evolução:
*     1.00   GB,JG,MV   03/04/2014 Início do desenvolvimento
*
***************************************************************************/

#include   <malloc.h>
#include   <stdio.h>

#define VALOR_OWN
#include "VALOR.H"
#undef VALOR_OWN

/***************************************************************************
*
*  $TC Tipo de dados: VAL Descritor do valor
*
*
*  $ED Descrição do tipo
*     Descreve a organização do valor
*
***************************************************************************/

   typedef struct tgValor {

      int NumCel;
            /* Número de Células */

      int Estado ;
            /* Estado do valor. 1 para resolvido e 0 para não resolvido. */

   } tpValor ;


/*****  Código das funções exportadas pelo módulo  *****/
```

```
/***************************************************************************
*
*  Função: VAL Criar Valor
*  ****/

   VAL_tpCondRet VAL_CriarValor( ptValor * pValor, int NumCel )
   {

      if ( *pValor != NULL )
      {
            VAL_DestruirValor( *pValor ) ;
      } /* if */

      (*pValor) = ( tpValor * ) malloc( sizeof( tpValor )) ;

      if ( *pValor == NULL )
      {
      return VAL_CondRetFaltouMemoria ;
      } /* if */

      (*pValor)->NumCel = NumCel ;
      (*pValor)->Estado = 0 ;

      return VAL_CondRetOK ;

   } /* Fim função: VAL Criar valor */

/***************************************************************************
*
*  Função: ARV Destruir valor
*  ****/

   VAL_tpCondRet VAL_DestruirValor( ptValor pValor )
   {

      if ( pValor != NULL )
      {
      free( pValor ) ;
      pValor = NULL ;
      } /* if */

      return VAL_CondRetOK;

   } /* Fim função: VAL Destruir valor */

/***************************************************************************
*
*  Função: VAL Alterar Número de Células
```

```
*  ****/

   VAL_tpCondRet VAL_AlterarNumCel( ptValor pValor, int NumCel )
   {

      if( pValor == NULL )
      {
            return VAL_CondRetValorNaoExiste ;
      } /* if */

      pValor->NumCel = NumCel;

      return VAL_CondRetOK ;

   } /* Fim função: VAL Alterar Número de Células */

/***************************************************************************
*
*  Função: VAL Alterar Estado
*  ****/

   VAL_tpCondRet VAL_AlterarEstado( ptValor pValor, int Estado )
   {

      if( pValor == NULL )
      {
            return VAL_CondRetValorNaoExiste ;
      } /* if */

      pValor->Estado = Estado;

      return VAL_CondRetOK ;

   } /* Fim função: VAL Alterar Estado */

/***************************************************************************
*
*  Função: VAL Obter estado
*  ****/

   VAL_tpCondRet VAL_ObterEstado( ptValor pValor, int * Estado )
   {

      if ( pValor == NULL )
      {
            return VAL_CondRetValorNaoExiste ;
      } /* if */
```

```
      *Estado = pValor->Estado ;

      return VAL_CondRetOK ;

   } /* Fim função: VAL Obter estado */

/***************************************************************************
*
*  Função: VAL Obter número de células
*  ****/

   VAL_tpCondRet VAL_ObterNumCel( ptValor pValor, int * NumCel )
   {

      if ( pValor == NULL )
      {
            return VAL_CondRetValorNaoExiste ;
      } /* if */

      *NumCel = pValor->NumCel ;

      return VAL_CondRetOK ;

   } /* Fim função: VAL Obter número de células */

/********** Fim do módulo de implementação: Módulo Valor **********/
```