```
#if ! defined( VALOR_ )
#define VALOR_
/***************************************************************************
*
*  $MCD Módulo de definição: VAL  Módulo Valor
*
*  Arquivo gerado:            VALOR.H
*  Letras identificadoras:    VAL
*
*  Nome da base de software:  Arcabouço para a automação de testes de
programas redigidos em C
*
*  Projeto: Trabalho 2 - Programação Modular
*  Autores: GB - Gustavo Bach
*           JG - João Lucas Gardenberg
*           MV - Michelle Valente
*
*  $HA Histórico de evolução:
*     Versão  Autor     Data              Observações
*     1     GB,JG,MV 28/mar/2013 início desenvolvimento
*
*  $ED Descrição do módulo
*     Esse módulo implementa funções para criar e explorar a
*           estrutura Valor.
*           O Valor contém um inteiro com o número de celulas que estão
*           preenchidas e outro inteiro informando qual o estado
*           daquele valor, resolvido ou não.
*
***************************************************************************/

#if defined( VALOR_OWN )
   #define VALOR_EXT
#else
   #define VALOR_EXT extern
#endif

/* Tipo referência para um valor */

typedef struct tgValor * ptValor ;


/***************************************************************************
*
*  $TC Tipo de dados: VAL Condições de retorno
*
*
*  $ED Descrição do tipo
*     Condições de retorno das funções do valor
*
```

```
***************************************************************************/

   typedef enum {

      VAL_CondRetOK = 0,
            /* Concluiu corretamente */

      VAL_CondRetNaoAchou ,
            /* Não encontrou o valor procurado */

      VAL_CondRetFaltouMemoria ,
            /* Faltou memória ao tentar criar um elemento de valor */

      VAL_CondRetValorNaoExiste
            /* Valor não existe */

   } VAL_tpCondRet ;

/***************************************************************************
*
*  $FC Função: VAL  &Criar Valor
*
*  $ED Descrição da função
*     Cria um novo valor, com o número de células enviado por parâmetro
*  e o Estado inicializado como 0.
*
*  $EP Parâmetros
*     pValor - ponteiro para o campo valor a ser criado
*     NumCel - número de celulas.
*
*  $FV Valor retornado
*     VAL_CondRetOK           - se criou sem problemas.
*     VAL_CondRetFaltouMemoria - se faltou memória para alocar o espaço do
valor.
*
*  Assertivas de entradas esperadas - NumCel é um inteiro.
*
*  Assertivas de saidas esperadas   - CondRetOk => pValor != NULL.
*                                   pValor->Numcel != NULL.
*                                   pValor->Estado = 0.
*                                   CondRetFaltouMemoria => pValor == NULL.
*
***************************************************************************/

   VAL_tpCondRet VAL_CriarValor( ptValor * pValor, int NumCel ) ;

/***************************************************************************
*
```

```
*  $FC Função: VAL  &Destruir Valor
*
*  $ED Descrição da função
*     Destroi o campo valor apontado pelo ponteiro.
*
*  $EP Parâmetros
*     pValor - ponteiro para o campo valor a ser criado
*
*  $FV Valor retornado
*     VAL_CondRetOK - se destruiu sem problemas.
*
*  Assertivas de entradas esperadas - pValor != NULL.
*
*  Assertivas de saidas esperadas   - pValor == NULL.
*
***************************************************************************/

  VAL_tpCondRet VAL_DestruirValor( ptValor pValor ) ;

/***************************************************************************
*
*  $FC Função: VAL  &Alterar NumCel
*
*  $ED Descrição da função
*     Altera o número de celulas contido no valor.
*
*  $EP Parâmetros
*     pValor - ponteiro para o campo valor a ser alterado
*     NumCel - número de células novo.
*
*  $FV Valor retornado
*     VAL_CondRetOK           - se alterou sem problemas.
*     VAL_CondRetValorNaoExiste - se o valor a ser alterado não existir.
*
*  Assertivas de entradas esperadas - NumCel é um inteiro.
*                                   pValor != NULL.
*
*  Assertivas de saidas esperadas   - CondRetOk => pValor != NULL.
*                                   pValor->Numcel = NumCel.
*                                   pValor->Estado não é alterado.
*                                   VAL_CondRetValorNaoExiste => pValor ==
NULL.
*
***************************************************************************/

   VAL_tpCondRet VAL_AlterarNumCel( ptValor pValor, int NumCel ) ;

/***************************************************************************
*
```

```
*  $FC Função: VAL  &Alterar Estado
*
*  $ED Descrição da função
*     Altera o estado corrente do valor.
*
*  $EP Parâmetros
*     pValor - ponteiro para o campo valor a ser alterado
*     Estado - int contendo 0 ou 1, sendo 1 para valor resolvido e 0 para
valor não resolvido.
*
*  $FV Valor retornado
*     VAL_CondRetOK           - se alterou sem problemas.
*     VAL_CondRetValorNaoExiste - se o valor a ser alterado não existir.
*
*  Assertivas de entradas esperadas - Estado é um inteiro.
*                                   pValor != NULL.
*
*  Assertivas de saidas esperadas   - CondRetOk => pValor != NULL.
*                                   pValor->Numcel não é alterado.
*                                   pValor->Estado == Estado.
*                                   VAL_CondRetValorNaoExiste => pValor ==
NULL.
*
***********************************************************************/

   VAL_tpCondRet VAL_AlterarEstado( ptValor pValor, int Estado ) ;

/***********************************************************************
*
*  $FC Função: VAL  &Obter Estado
*
*  $ED Descrição da função
*     Obter o estado atual daquele valor, sendo 1 para resolvido e 0 para não
resolvido.
*
*  $EP Parâmetros
*     pValor - ponteiro para o campo valor
*     Estado - parâmetro que receberá o estado contido no valor.
*                 Este parâmetro é passado por referência.
*
*  $FV Valor retornado
*     VAL_CondRetOK           - se obteve sem problemas.
*     VAL_CondRetValorNaoExiste - se o valor a ser utilizado não existir.
*
*  Assertivas de entradas esperadas - pValor != NULL.
*
*  Assertivas de saidas esperadas   - CondRetOk => pValor não é alterado.
*                                   Estado == pValor->Estado.
```

```
*                                   VAL_CondRetValorNaoExiste => pValor ==
NULL.
*
***********************************************************************/

   VAL_tpCondRet VAL_ObterEstado( ptValor pValor, int * Estado ) ;

/***********************************************************************
*
*  $FC Função: VAL  &Obter NumCel
*
*  $ED Descrição da função
*     Obter o número de células.
*
*  $EP Parâmetros
*     pValor - ponteiro para o campo valor
*     NumCel - parâmetro que receberá o número de celulas contido no valor.
*                 Este parâmetro é passado por referência.
*
*  $FV Valor retornado
*     VAL_CondRetOK           - se obteve sem problemas.
*     VAL_CondRetValorNaoExiste - se o valor a ser utilizado não existir.
*
*  Assertivas de entradas esperadas - pValor != NULL.
*
*  Assertivas de saidas esperadas   - CondRetOk => pValor não é alterado.
      *                                    NumCel == pValor->NumCel.
*                                   VAL_CondRetValorNaoExiste => pValor ==
NULL.
*
***********************************************************************/

   VAL_tpCondRet VAL_ObterNumCel( ptValor pValor, int * NumCel ) ;

#undef VAL_EXT

/********** Fim do módulo de definição: VAL  Módulo Valor **********/

#else
#endif
```