```
/***************************************************************************
*  $MCI Módulo de implementação: Módulo de teste específico
*
*  Arquivo gerado:              TESTCEL.C
*  Letras identificadoras:      TCEL
*
*  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:
*     Versão  Autor      Data          Observações
*     1.00    GB,JG,MV   12/abr/2014   Início do desenvolvimento
*
*  $ED Descrição do módulo
*
*  $EIU Interface com o usuário pessoa
*     Comandos de teste específicos para testar o módulo árvore:
*
*     "=criar" <Int> <Int>
*                     - chama a função CEL_CriarCelula( ).
*                       Obs. notação: O 1o <Int> é o número da célula.
*                       Obs. notação: O 2o <Int> é a condição de retorno
esperada.
*     "=destruir" <Int> <Int>
*                     - chama a função CEL_DestruirCelula( ).
*                       Obs. notação: O 1o <Int> é o número da célula.
*                       Obs. notação: O 2o <Int> é a condição de retorno
esperada.
*     "=obteratual" <Int> <Int> <Int>
*                     - chama a função CEL_ObterEstadoAtual( ) e compara o
valor
*                       retornado com o 2o <Int>.
*                       Obs. notação: O 1o <Int> é o número da célula.
*                       Obs. notação: O 2o <Int> é o estado atual esperado.
*                       Obs. notação: O 3o <Int> é a condição de retorno
esperada.
*     "=obtercorreto" <Int> <Int> <Int>
*                     - chama a função CEL_ObterEstadoCorreto( ) e compara o
valor
*                       retornado com o 2o <Int>.
*                       Obs. notação: O 1o <Int> é o número da célula.
*                       Obs. notação: O 2o <Int> é o estado correto esperado.
*                       Obs. notação: O 3o <Int> é a condição de retorno
esperada.
*     "=alteraratual" <Int> <Int>
```

```
*                    - chama a função CEL_AlterarEstadoAtual( ).
*                      Obs. notação: O 1o <Int> é o número da célula.
*                      Obs. notação: O 2o <Int> é a condição de retorno
esperada.
*     "=alterarcorreto" <Int> <Int>
*                    - chama a função CEL_AlterarEstadoCorreto( ).
*                      Obs. notação: O 1o <Int> é o número da célula.
*                      Obs. notação: O 2o <Int> é a condição de retorno
esperada.
*     "=comparar" <Int> <Int> <Int>
*                    - chama a função CEL_CompararEstados( ) e compara o
valor
*                      retornado com o 2o <Int>.
*                      Obs. notação: O 1o <Int> é o número da célula.
*                      Obs. notação: O 2o <Int> é a comparação esperada.
*                      0 = Iguais; 1 = Diferentes.
*                      Obs. notação: O 3o <Int> é a condição de retorno
esperada.
*
***************************************************************************/

#include    <string.h>
#include    <stdio.h>

#include    "TST_ESPC.H"

#include    "generico.h"
#include    "lerparm.h"

#include    "CELULA.H"

#define           NUM_CELULAS            5

/* Tabela dos nomes dos comandos de teste específicos */

#define     CRIAR_CEL_CMD         "=criar"
#define     DESTRUIR_CEL_CMD      "=destruir"
#define     OBTER_ATUAL_CMD       "=obteratual"
#define     OBTER_CORRETO_CMD     "=obtercorreto"
#define     ALTERAR_ATUAL_CMD     "=alteraratual"
#define     ALTERAR_CORRETO_CMD   "=alterarcorreto"
#define     COMPARAR_CEL_CMD      "=comparar"

/*****  Declarações de variáveis  *****/

        ptCelula VetCelula [ NUM_CELULAS ] ;
              /* Vetor contendo os ponteiros para as células */
```

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


/***************************************************************************
*
*  $FC Função: TCEL Efetuar operações de teste específicas para célula
*
*  $ED Descrição da função
*     Efetua os diversos comandos de teste específicos para o módulo
*     célula sendo testado.
*
*  $EP Parâmetros
*     $P ComandoTeste - String contendo o comando
*
*  $FV Valor retornado
*     Ver TST_tpCondRet definido em TST_ESPC.H
*
***************************************************************************/

   TST_tpCondRet TST_EfetuarComando( char * ComandoTeste )
   {

      CEL_tpCondRet CondRetObtido   = CEL_CondRetOK ;
      CEL_tpCondRet CondRetEsperada = CEL_CondRetFaltouMemoria ;
                                      /* inicializa para qualquer coisa */
        int NumCelula ;
        int ValorEsperado = 0 ;
      int ValorObtido   = 0 ;

        int  NumLidos = -1 ;

        TST_tpCondRet Ret ;

        /* Testar CEL Criar célula */

           if( strcmp( ComandoTeste, CRIAR_CEL_CMD ) == 0 )
             {

            NumLidos = LER_LerParametros( "ii",
                                                  &NumCelula,

&CondRetEsperada ) ;
                  if( NumLidos != 2 )
                  {
                        return TST_CondRetParm ;
                  } /* if */

                  CondRetObtido = CEL_CriarCelula( &VetCelula[ NumCelula ] ) ;
```

```
                return TST_CompararInt( CondRetEsperada,
                                        CondRetObtido,
                                        "Retorno errado ao criar célula."
) ;

             } /* fim ativa: Testar CEL Criar célula */

        /* Testar CEL Destruir célula */

           else if( strcmp( ComandoTeste, DESTRUIR_CEL_CMD ) == 0 )
             {

            NumLidos = LER_LerParametros( "ii",
                                                &NumCelula,

&CondRetEsperada ) ;
                if( NumLidos != 2 )
                {
                      return TST_CondRetParm ;
                } /* if */

                CondRetObtido = CEL_DestruirCelula( &VetCelula[ NumCelula ]
) ;

                return TST_CompararInt( CondRetEsperada,
                                        CondRetObtido,
                                        "Retorno errado ao destruir
célula." ) ;

             } /* fim ativa: Testar CEL Destruir célula */

        /* Testar CEL Obter estado atual da célula */

           else if( strcmp( ComandoTeste, OBTER_ATUAL_CMD ) == 0 )
             {

            NumLidos = LER_LerParametros( "iii",
                                                &NumCelula,
                                                            &ValorEsperado,

&CondRetEsperada ) ;
                  if ( NumLidos != 3 )
            {
               return TST_CondRetParm ;
            } /* if */
```

```
                CondRetObtido = CEL_ObterEstadoAtual( &VetCelula[ NumCelula
],
                                                      &ValorObtido

                Ret = TST_CompararInt( CondRetEsperada,
                                       CondRetObtido,
                                              "Retorno errado ao obter
estado atual." ) ;

                if ( Ret != TST_CondRetOK )
            {
               return Ret ;
            } /* if */

                return TST_CompararInt( ValorEsperado,
                                        ValorObtido,
                                                 "Conteúdo do estado
atual da célula está errado." ) ;

             } /* fim ativa: Testar CEL Obter estado atual da célula */

        /* Testar CEL Obter estado correto da célula */

           else if( strcmp( ComandoTeste, OBTER_CORRETO_CMD ) == 0 )
             {

            NumLidos = LER_LerParametros( "iii",
                                                &NumCelula,
                                                           &ValorEsperado,

&CondRetEsperada ) ;
                if ( NumLidos != 3 )
            {
               return TST_CondRetParm ;
            } /* if */

                CondRetObtido = CEL_ObterEstadoCorreto( &VetCelula[
NumCelula ],
                                                        &ValorObtido

                Ret = TST_CompararInt( CondRetEsperada,
                                       CondRetObtido,
                                              "Retorno errado ao obter
estado correto da célula." ) ;

                if ( Ret != TST_CondRetOK )
            {
               return Ret ;
```

```
            } /* if */

                  return TST_CompararInt( ValorEsperado,
                                          ValorObtido,
                                                    "Conteúdo do estado
correto da célula está errado." ) ;

             } /* fim ativa: Testar CEL Obter estado correto da célula */

        /* Testar CEL Alterar estado atual da célula */

           else if( strcmp( ComandoTeste, ALTERAR_ATUAL_CMD ) == 0 )
             {

            NumLidos = LER_LerParametros( "ii",
                                                  &NumCelula,

&CondRetEsperada ) ;
                  if ( NumLidos != 2 )
            {
               return TST_CondRetParm ;
            } /* if */

                  CondRetObtido = CEL_AlterarEstadoAtual( &VetCelula[
NumCelula ] ) ;

                  return TST_CompararInt( CondRetEsperada,
                                          CondRetObtido,
                                                    "Retorno errado ao
alterar estado atual da célula." ) ;

             } /* fim ativa: Testar CEL Alterar estado atual da célula */

        /* Testar CEL Alterar estado correto da célula */

           else if( strcmp( ComandoTeste, ALTERAR_CORRETO_CMD ) == 0 )
             {

            NumLidos = LER_LerParametros( "ii",
                                                  &NumCelula,

&CondRetEsperada ) ;
                  if ( NumLidos != 2 )
            {
               return TST_CondRetParm ;
            } /* if */
```

```
                  CondRetObtido = CEL_AlterarEstadoCorreto( &VetCelula[
NumCelula ] ) ;

                  return TST_CompararInt( CondRetEsperada,
                                          CondRetObtido,
                                                   "Retorno errado ao
alterar estado correto da célula." ) ;

              } /* fim ativa: Testar CEL Alterar estado correto da célula */

        /* Testar CEL Comparar estados da célula */

           else if( strcmp( ComandoTeste, COMPARAR_CEL_CMD ) == 0 )
             {

            NumLidos = LER_LerParametros( "iii",
                                                 &NumCelula,
                                                             &ValorEsperado,

&CondRetEsperada ) ;
                  if ( NumLidos != 3 )
            {
               return TST_CondRetParm ;
            } /* if */

                  CondRetObtido = CEL_CompararEstados( &VetCelula[ NumCelula
],
                                                       &ValorObtido

                  Ret = TST_CompararInt( CondRetEsperada,
                                         CondRetObtido,
                                                 "Retorno errado ao
comparar estados da célula." ) ;

                  if ( Ret != TST_CondRetOK )
            {
               return Ret ;
            } /* if */

                  return TST_CompararInt( ValorEsperado,
                                          ValorObtido,
                                                   "O valor retornado pela
comparação dos estados da célula está errado." ) ;

              } /* fim ativa: Testar CEL Comparar estados da célula */

   } /* Fim função: TCEL Efetuar operações de teste específicas para célula
*/
```

```
/********** Fim do módulo de implementação: Módulo de teste específico
**********/
```