```
#if ! defined( CELULA_ )
#define CELULA_
/***************************************************************************
*
*  $MCD Módulo de definição: CEL  Célula
*
*  Arquivo gerado:            CELULA.h
*  Letras identificadoras:    CEL
*
*  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
*     1.00  GB,JG,MV   12/abr/2014   Correções das condições de retorno
*                                    e dos parâmetros por referência.
*     1.00  GB,JG,MV   03/abr/2014   Início do desenvolvimento.
*
*  $ED Descrição do módulo
*     Este módulo implementa um conjunto simples de funções para criar e
*     explorar células.
*     A célula possui um Estado Atual, que diz se ela está preenchida ou vazia
*     e um Estado Correto, que diz se era pra ela estar preenchida ou vazia.
*     Ao iniciar a execução do programa não existe célula.
*     A célula é inicializada com os estados Atual e Correto vazios.
*
***************************************************************************/

#if defined( CELULA_OWN )
   #define CELULA_EXT
#else
   #define CELULA_EXT extern
#endif

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

/* Tipo referência para uma célula */

typedef struct tgCelula * ptCelula ;


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

```
*  $TC Tipo de dados: CEL Condições de retorno
*
*  $ED Descrição do tipo
*     Condições de retorno das funções da célula
*
***************************************************************************/

   typedef enum {

      CEL_CondRetOK = 0,
            /* Concluiu corretamente */

      CEL_CondRetFaltouMemoria = 1,
            /* Faltou memória ao tentar criar uma célula */

      CEL_CondRetCelulaNaoExiste = 2,
            /* Célula não existe */

      CEL_CondRetEstadoInvalido = 3
            /* Obteve estado diferente de 1 ou 0 (preenchida ou
                                                            vazia) */

   } CEL_tpCondRet ;


/***************************************************************************
*
*  $FC Função: CEL  Criar célula
*
*  $ED Descrição da função
*     Cria uma nova célula, com os estados atual e correto vazios.
*     Caso já exista uma célula, esta será destruída.
*
*  $EP Parâmetros
*     $P pCelula - ponteiro para a célula a ser criada.
*
*  $FV Valor retornado
*     CEL_CondRetOK           - se criou sem problemas.
*     CEL_CondRetFaltouMemoria - se faltou memória para alocar o espaço da
célula.
*
*  Assertivas de entradas esperadas - Nenhuma.
*
*  Assertivas de saidas esperadas   - CEL_CondRetOK => pCelula != NULL.
*                                  CEL_CondRetFaltouMemoria => pCelula ==
NULL.
*                                  pCelula->EstadoAtual == 0.
*                                  pCelula->EstadoCorreto == 0.
```

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

   CEL_tpCondRet CEL_CriarCelula( ptCelula * pCelula ) ;


/***************************************************************************
*
*  $FC Função: CEL  Destruir célula
*
*  $ED Descrição da função
*     Destrói a célula corrente.
*     Faz nada caso a célula corrente não exista.
*
*  $EP Parâmetros
*     $P pCelula - ponteiro para a célula a ser destruída.
*
*  $FV Valor retornado
*     CEL_CondRetOK - se destruiu sem problemas.
*
*  Assertivas de entradas esperadas - pCelula != NULL.
*
*  Assertivas de saidas esperadas   - pCelula == NULL.
*
***************************************************************************/

   CEL_tpCondRet CEL_DestruirCelula( ptCelula pCelula ) ;


/***************************************************************************
*
*  $FC Função: CEL  Obter estado atual da célula
*
*  $EP Parâmetros
*     $P pCelula - ponteiro para a célula a ser analisada.
*     $P Estado - é o parâmetro que receberá o estado atual da célula.
*                 Este parâmetro é passado por referência.
*                 Se receber 0, a célula está vazia.
*                 Se receber 1, a célula está preenchida.
*                 Se receber -1, o estado está inválido.
*
*  $FV Valor retornado
*     CEL_CondRetOK           - se obteve o estado atual sem problemas.
*     CEL_CondRetCelulaNaoExiste - se a célula a ser utilizada não existir.
*     CEL_CondRetEstadoInvalido  - se o estado atual estiver inválido.
*
*  Assertivas de entradas esperadas - pCelula != NULL.
*                                  pCelula->EstadoAtual == 0 || pCelula-
>EstadoAtual
```

```
*                                  == 1.
*
*  Assertivas de saidas esperadas   - CEL_CondRetOK => Estado == 0 || Estado
== 1.
*                                  CEL_CondRetCelulaNaoExiste => pCelula ==
NULL.
*                                  CEL_CondRetEstadoInvalido => Estado == -1.
*                                  Estado == pCelula->EstadoAtual.
*                                  pCelula->EstadoAtual não é alterado.
*                                  pCelula->EstadoCorreto não é alterado.
*
***********************************************************************/

   CEL_tpCondRet CEL_ObterEstadoAtual( ptCelula pCelula,
                                       int * Estado      ) ;


/***********************************************************************
*
*  $FC Função: CEL  Obter estado correto da célula
*
*  $EP Parâmetros
*     $P pCelula - ponteiro para a célula a ser analisada.
*     $P Estado - é o parâmetro que receberá o estado correto da célula.
*                 Este parâmetro é passado por referência.
*                 Se receber 0, o estado correto da célula é vazia.
*                 Se receber 1, o estado correto da célula é preenchida.
*                 Se receber -1, o estado está inválido.
*
*  $FV Valor retornado
*     CEL_CondRetOK           - se obteve o estado correto sem problemas.
*     CEL_CondRetCelulaNaoExiste - se a célula a ser utilizada não existir.
*     CEL_CondRetEstadoInvalido  - se o estado correto estiver inválido.
*
*  Assertivas de entradas esperadas - pCelula != NULL.
*                                  pCelula->EstadoCorreto == 0 || pCelula-
>EstadoCorreto
*                                  == 1.
*
*  Assertivas de saidas esperadas   - CEL_CondRetOK => Estado == 0 || Estado
== 1.
*                                  CEL_CondRetCelulaNaoExiste => pCelula ==
NULL.
*                                  CEL_CondRetEstadoInvalido => Estado == -1.
*                                  Estado == pCelula->EstadoCorreto.
*                                  pCelula->EstadoAtual não é alterado.
*                                  pCelula->EstadoCorreto não é alterado.
*
*
```

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

   CEL_tpCondRet CEL_ObterEstadoCorreto( ptCelula pCelula,
                                         int * Estado      ) ;


/***************************************************************************
*
*  $FC Função: CEL  Alterar estado atual da célula
*
*  $ED Descrição da função
*     Altera o estado atual da célula.
*     Se estiver vazia, passa a ser preenchida.
*     Se estiver preenchida, passa a ser vazia.
*
*  $EP Parâmetros
*     $P pCelula - ponteiro para a célula a ser alterada.
*
*  $FV Valor retornado
*     CEL_CondRetOK           - se alterou o estado atual sem problemas.
*     CEL_CondRetCelulaNaoExiste - se a célula a ser alterada não existir.
*     CEL_CondRetEstadoInvalido  - se o estado atual estiver inválido.
*
*  Assertivas de entradas esperadas - pCelula != NULL.
*                                  pCelula->EstadoAtual == 0 || pCelula-
>EstadoAtual
*                                  == 1.
*
*  Assertivas de saidas esperadas   - CEL_CondRetOK => (pCelula->EstadoAtual
== 0 =>
*                                  pCelula->EstadoAtual == 1) || (pCelula-
>EstadoAtual
*                                  == 1 => pCelula->EstadoAtual == 0).
*                                  CEL_CondRetCelulaNaoExiste => pCelula ==
NULL.
*                                  CEL_CondRetEstadoInvalido => pCelula-
>EstadoAtual
*                                  != 1 && pCelula->EstadoAtual != 0.
*                                  pCelula->EstadoCorreto não é alterado.
*
***************************************************************************/

   CEL_tpCondRet CEL_AlterarEstadoAtual( ptCelula pCelula ) ;


/***************************************************************************
*
*  $FC Função: CEL  Alterar estado correto da célula
*
```

```
*  $ED Descrição da função
*     Altera o estado correto da célula.
*     Se o estado correto é preenchida, este passa a ser vazia.
*     Se o estado correto é vazia, este passa a ser preenchida.
*
*  $EP Parâmetros
*     $P pCelula - ponteiro para a célula a ser alterada.
*
*  $FV Valor retornado
*     CEL_CondRetOK           - se alterou o estado correto sem problemas.
*     CEL_CondRetCelulaNaoExiste - se a célula a ser alterada não existir.
*     CEL_CondRetEstadoInvalido  - se o estado correto estiver inválido.
*
*  Assertivas de entradas esperadas - pCelula != NULL.
*                                   pCelula->EstadoCorreto == 0 || pCelula-
>EstadoCorreto
*                                   == 1.
*
*  Assertivas de saidas esperadas   - CEL_CondRetOK => (pCelula-
>EstadoCorreto == 0 =>
*                                   pCelula->EstadoCorreto == 1) || (pCelula-
>EstadoCorreto
*                                   == 1 => pCelula->EstadoCorreto == 0).
*                                   CEL_CondRetCelulaNaoExiste => pCelula ==
NULL.
*                                   CEL_CondRetEstadoInvalido => pCelula-
>EstadoCorreto
*                                   != 1 && pCelula->EstadoCorreto != 0.
*                                   pCelula->EstadoAtual não é alterado.
*
***********************************************************************/

   CEL_tpCondRet CEL_AlterarEstadoCorreto( ptCelula pCelula ) ;


/***********************************************************************
*
*  $FC Função: CEL  Comparar estados da célula
*
*  $ED Descrição da função
*     Compara os estados atual e correto da célula.
*
*  $EP Parâmetros
*     $P pCelula - ponteiro para a célula a ser analisada.
*     $P Comparacao - é o parâmetro que receberá a comparação.
*                     Este parâmetro é passado por referência.
*                     Se receber 0, os estados estão iguais.
*                     Se receber 1, os estados estão diferentes.
*                     Se receber -1, algum dos estados está inválido.
```

```
*
*  $FV Valor retornado
*     CEL_CondRetOK           - se a comparação ocorreu sem problemas.
*     CEL_CondRetCelulaNaoExiste - se a célula a ser utilizada não existir.
*     CEL_CondRetEstadoInvalido  - se algum dos estados estiver inválido.
*
*  Assertivas de entradas esperadas - pCelula != NULL.
*                                   pCelula->EstadoAtual == 0 || pCelula-
>EstadoAtual
*                                   == 1.
*
*  Assertivas de saidas esperadas   - CEL_CondRetOK => Comparacao == 0 ||
Comparacao == 1.
*                                   CEL_CondRetCelulaNaoExiste => pCelula ==
NULL.
*                                   CEL_CondRetEstadoInvalido => Comparacao
== -1.
*                                   pCelula->EstadoAtual não é alterado.
*                                   pCelula->EstadoCorreto não é alterado.
*
***********************************************************************/

   CEL_tpCondRet CEL_CompararEstados( ptCelula pCelula,
                                      int * Comparacao ) ;


/********** Fim do módulo de definição: CEL  Célula **********/

#else
#endif
```