```
#if ! defined( MATRIZ_ )
   #define MATRIZ_
/***************************************************************************
*
*  $MCD Módulo de definição: MAT  Matriz de listas
*
*  Arquivo gerado:            MATRIZ.h
*  Letras identificadoras:    MAT
*
*  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
*     2.00  GB,JG,MV   11/abr/2013   reformulação de todas as funções
*     1.00  GB,JG,MV   28/mar/2013   início desenvolvimento
*
*  $ED Descrição do módulo
*
*
***************************************************************************/


#if defined( MATRIZ_OWN )
   #define MATRIZ_EXT
#else
   #define MATRIZ_EXT extern
#endif

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

/* Tipo referência para uma matriz */

typedef struct tgMatriz * ptMatriz ;


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

```
   typedef enum {

      MAT_CondRetOK,
            /* Concluiu corretamente */

      MAT_CondRetMatrizNaoExiste,
            /* Não concluiu corretamente */

      MAT_CondRetMatrizVazia,
            /* A matriz não contém elementos */

      MAT_CondRetNaoAchou,
            /* Não encontrou o valor procurado */

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

      MAT_CondRetFimColunas,
            /* Foi atingido o fim das colunas da matriz */

      MAT_CondRetFimLinhas
            /* Foi atingido o fim das linhas da matriz */

   } MAT_tpCondRet ;


/***************************************************************************
*
*  $FC Função: MAT  Criar matriz
*
*  $ED Descrição da função
*     Cria uma matriz genérica.
*     Os possíveis tipos são desconhecidos a priori.
*     A tipagem é implícita.
*     Não existe identificador de tipo associado à matriz.
*
*  $EP Parâmetros
*     $P pMatriz - ponteiro para a matriz a ser criada.
*                  Este ponteiro será utilizado pelas funções
*                  que manipulem esta matriz.
*     $P Altura  - define qual vai ser a altura da matriz.
*     $P Largura - define qual vai ser a largura da matriz.
*
*  $FV Valor retornado
*     MAT_CondRetOK           - se criou sem problemas.
*     MAT_CondRetFaltouMemoria - se faltou memória para alocar o espaço da
matriz.
```

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

   MAT_tpCondRet MAT_CriarMatriz( ptMatriz * pMatriz, int Altura, int Largura
);


/***********************************************************************
*
*  $FC Função: MAT  Destruir matriz
*
*  $ED Descrição da função
*     Destrói a matriz fornecida.
*     O parâmetro ponteiro para a matriz não é modificado.
*     Se ocorrer algum erro durante a destruição, a matriz resultará
*     estruturalmente incorreta.
*     OBS. não existe previsão para possíveis falhas de execução.
*
*  $EP Parâmetros
*     $P pMatriz - ponteiro para a matriz a ser destruída.
*
*  $FV Valor retornado
*     MAT_CondRetOK           - se destruiu sem problemas
*     MAT_CondRetMatrizNaoExiste - se a matriz a ser destruída não existir.
*
***********************************************************************/

   MAT_tpCondRet MAT_DestruirMatriz( ptMatriz pMatriz ) ;


/***********************************************************************
*
*  $FC Função: MAT  Inserir valor no elemento
*
*  $ED Descrição da função
*     Insere novo elemento na posição especificada.
*
*  $EP Parâmetros
*     $P pMatriz   - ponteiro para a matriz a ser alterada.
*     $P pElemento - ponteiro para o elemento que será inserido.
*     $P Linha     - linha onde o elemento será inserido.
*                  O número tem que ser maior que 0 e menor ou igual
*                  à altura da matriz.
*     $P Coluna    - coluna onde o elemento será inserido.
*                  O número tem que ser maior que 0 e menor ou igual
*                  à coluna da matriz.
*
*  $FV Valor retornado
*     MAT_CondRetOK           - se inseriu o elemento sem problemas.
```

```
*     MAT_CondRetMatrizNaoExiste - se a matriz a ser alterada não existir.
*     MAT_CondRetMatrizVazia  - se a matriz a ser alterada estiver vazia.
*     MAT_CondRetFimLinhas    - se o parâmetro Linha passado for maior do que
*                                 o número de linhas da matriz a ser
alterada.
*     MAT_CondRetFimColunas   - se o parâmetro Coluna passado for maior do que
*                                 o número de colunas da matriz a ser
alterada.
*
***********************************************************************/

   MAT_tpCondRet MAT_InserirValor( ptMatriz pMatriz, void * pElemento, int
Linha , int Coluna ) ;


/***********************************************************************
*
*  $FC Função: MAT  Obter valor do elemento
*
*  $ED Descrição da função
*     Obtém o valor do elemento que está na posição especificada.
*
*  $EP Parâmetros
*     $P pMatriz - ponteiro para a matriz onde o elemento se encontra.
*     $P Linha   - linha onde o elemento se encontra.
*     $P Coluna  - coluna onde o elemento se encontra.
*     $P pValor  - ponteiro que receberá a referência para o valor contido
*                  no elemento.
*
*  $FV Valor retornado
*     MAT_CondRetOK           - se obteve o valor do elemento sem problemas.
*     MAT_CondRetMatrizNaoExiste - se a matriz a ser utilizada não existir.
*     MAT_CondRetMatrizVazia  - se a matriz a ser utilizada estiver vazia.
*     MAT_CondRetFimLinhas    - se o parâmetro Linha passado for maior do que
*                                 o número de linhas da matriz utilizada.
*     MAT_CondRetFimColunas   - se o parâmetro Coluna passado for maior do que
*                                 o número de colunas da matriz utilizada.
*
***********************************************************************/

   MAT_tpCondRet MAT_ObterValor( ptMatriz pMatriz, int Linha, int Coluna,
void ** pValor ) ;


#undef MAT_EXT

/********** Fim do módulo de definição: MAT  Matriz de listas **********/
```

```
#else
#endif
```