```
/*============================================================================*/
/*   MODULE                    ERROR.C                                        */
/*============================================================================*/
/* FUNCTION    This module implements a common function for error reporting
 *             for the super-project : "FIND".
 *
 * SYSTEM      Standard (ANSI/ISO) C.
 *             Tested on PC/MS DOS 5.0
 *
 * SEE ALSO    ERROR.H (and application modules such as: BOOL.C, AC.C)
 *
 * PROGRAMMER  Allan Dystrup
 *
 * COPYRIGHT   (c) Allan Dystrup
 *
 * VERSION     $Header: d:/cwk/kf/error/RCS/error.c 1.1 92/10/25 17:13:30
 *             Allan_Dystrup Exp Locker: Allan_Dystrup $
 *             ---------------------------------------------------------------
 *             $Log:    error.c $
 *             Revision 1.1  92/10/25  17:13:30  Allan_Dystrup
 *             Initial revision
 *
 *============================================================================*/



/*============================================================================*/
/*                             Includes                                       */
/*============================================================================*/

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

#include "error.h"



/*----------------------------------------------------------------------------*/
/*                         Error Messages                                     */
/*----------------------------------------------------------------------------*/
/* Error message format :
 *    error header       - severity code : MODULE[function(s)] : unique tag
 *    error description  - what went wrong
 *    error correction   - what to to
 */
PRIVATE char *mEARG[] = {
    "\n  STOP : [BOOL|BM] : E[%03d]ARG\n",
    "  Argumentfejl i inddata : forkert aktuel parameter i funktionskald\n",
    "  Programm›r: L‘s manualside. --- Bruger: Underret Kommunedata.\n\n", 0};
PRIVATE char *mELEX[] = {
    "\n  STOP : BOOL[bScan] : E[%03d]LEX\n",
    "  Skrivefejl i inddata : ikke tilladt eller manglende tegn i s›geudtryk\n",
    "  Ret s›geudtryk og gentag s›gning\n\n", 0};
PRIVATE char *mESYN[] = {
    "\n  STOP : BOOL[pzParse] : E[%03d]SYN\n",
    "  Syntaksfejl i inddata : forkert opbygning af s›geudtryk\n",
    "  Ret s›geudtryk og gentag s›gning\n\n", 0};
PRIVATE char *mETAB[] = {
    "\n  STOP : BOOL[vEmit|iSymInsert] : E[%03d]TAB\n",
    "  Programfejl i tabeller : ikke plads nok i programmets datastrukturer\n",
    "  Underret Kommunedata, - pr›v med kortere s›geudtryk\n\n", 0};
PRIVATE char *mETOK[] = {
    "\n  STOP : BOOL[vEmit|fInterpret] : E[%03d]TOK\n",
    "  Programfejl i intermedi‘r kode : ukendt kompilersymbol (token)\n",
```

```
    "  Underret Kommunedata.\n\n", 0};
PRIVATE char *mEMEM[] = {
    "\n  STOP : [AC|BM] : E[%03d]MEM\n",
    "  Intern fejl i lagerallokering : ikke nok dynamisk lager (heap)\n",
    "  Underret Kommunedata, - pr›v med kortere s›geudtryk\n\n", 0};


/* Point unique ErrorCodes in enum error to appropriate ErrorMessages */
/* OBS: Several ErrorCodes may "share" the same type of ErrorMessage. */
PRIVATE char **Errmsgs[] = {   /* Global E#     Local E# */
    mEARG,                     /* E[000]ARG  =  EARG000  */
    mEARG,                     /* E[001]ARG  =  EARG001  */
    mEARG,                     /* E[002]ARG  =  EARG002  */
    mEARG,                     /* E[003]ARG  =  EARG003  */
    mEARG,                     /* E[004]ARG  =  EARG004  */
    mELEX,                     /* E[005]LEX  =  ELEX000  */
    mESYN,                     /* E[006]SYN  =  ESYN000  */
    mETAB,                     /* E[007]TAB  =  ETAB000  */
    mETAB,                     /* E[008]TAB  =  ETAB001  */
    mETAB,                     /* E[009]TAB  =  ETAB002  */
    mETOK,                     /* E[010]TOK  =  ETOK000  */
    mETOK,                     /* E[011]TOK  =  ETOK001  */
    mEMEM,                     /* E[012]MEM  =  EMEM000  */
    mEMEM,                     /* E[013]MEM  =  EMEM001  */
    mEMEM,                     /* E[014]MEM  =  EMEM002  */
    mEMEM,                     /* E[015]MEM  =  EMEM003  */
    mEMEM,                     /* E[016]MEM  =  EMEM004  */
    mEMEM,                     /* E[017]MEM  =  EMEM005  */
    mEMEM,                     /* E[018]MEM  =  EMEM006  */
    mEMEM,                     /* E[019]MEM  =  EMEM007  */
    mEMEM                      /* E[020]MEM  =  EMEM008  */
};



#ifdef MAIN
/*+2 MODULE ERROR.C ========================================================*/
/*   NAME   00                    main                                      */
/*== SYNOPSIS ==============================================================*/
int
main()
{
/* DESCRIPTION
 *    Test driver for module error.c :
 *    Dump all ErrorMessages to stderr, using vError() for printout.
 *-2*/

   ERRNUM   errno;     /* For stepping through enumation : ERRNUM */
   BYTE     *msg;      /* For pointing to corresp. err.mess., cf. Errmsg[] */


   printf("\n======= Testudskrift af samtlige fejlmeddelelser =======\n\n");

   for (errno = 0; errno <= ERRMAX; errno++) {
       msg = Errmsgs[(int) errno][0] + 3;
       if ( *msg != '#' ) {    /* Mark message in 1. byte of severity code */
          *msg = '#';          /* (dump only once & prevent EXIT_FAILURE). */
          vError(errno, "TEST");
       }
   }

}
#endif /* MAIN */
```

```
/*+2 MODULE ERROR.C ============================================================*/
/*   NAME   01                     vError                                       */
/*== SYNOPSIS ================================================================*/
PUBLIC void
vError( ERRNUM type,       /* Error type, cf. enum errors */
        char *param )      /* Further description of error location */
{
/* DESCRIPTION
 *    Simple error management:
 *      1. Report error
 *      2. If fatal, return exit status 'failure' to parent process
 *-2*/
    int       i = 0;

    /* 1: Report error to stderr ... */
    fprintf(stderr, "\n\a=>%s", param);                   /* Pinpoint error */

    fprintf(stderr, Errmsgs[(int) type][i++], type);      /* Error header   */

    while (Errmsgs[(int) type][i])                      /* Description    */
       fputs(Errmsgs[(int) type][i++], stderr);           /* & correction   */


    /* If fatal (ie. severety code STOP), exit ' failure' */
    if (strstr(Errmsgs[(int) type][0], "STOP"))           /* Fatal => quit  */
       exit(EXIT_FAILURE);                                /* cf. stdlib.h   */

} /* END function vError() */



/* END module ERROR.C                                                         */
/*============================================================================*/
```