```
/*


          FICHIER PRIMITIVES DE BASE (1ere partie)
          ----------------------------------------

                    (version curses.h )

                    multi-fenetrage

                   (  O.B.J.E.T )

*/

#include <stdio.h>
#include<stdlib.h>
#include <signal.h>
#include <ctype.h>
#include <curses.h>
#include <string.h>
#include "pconstante.h"
#include <errno.h>

#include "prototype.h"

#ifndef ACS_URCORNER
#define ACS_VLINE '|'
#define ACS_HLINE '-'
#define ACS_ULCORNER '.'
#define ACS_URCORNER '.'
#define ACS_LLCORNER '.'
#define ACS_LRCORNER '.'
#endif

#define CORRECTION '\6'       /* activation mode correction par ^F */

/*----------- caracteristiques associees au termcap ------------*/

static char t_f[512]       ;  /* liste des codes touches fonct. et depl*/

void f_option(void);

/*---------------codification attributs visualisation------------------

   attributs codes sous forme d'un caractere :

      -i : inverse video
      -u : souslignement
      -h : demi intensite
      -c : clignotement

------------------------------------------------------------------------*/

/* char *getenv(char *),*tgetstr(char *,char *),*tgoto(char *,int,int); */

/*------------------- variable pour mode correction ------------------*/

static short correction; /* =VRAI si mode correction autorise
                     =FAUX sinon */
```

```
/*-----------------------------------------------------------------*/

void invalide_tf(char t)

/* rend inutilisable la fleche ou la touche fonction t */

/*-----------------------------------------------------------------------*/

void valide_tf(char t)

/* rend utilisable la fleche ou la touche fonction t
   si t = -1, toutes les touches fonctions et fleches sont renduees utili-
              sables
*/

/*-----------------------------------------------------------*/

short nbligne_ecran( void )

/* retourne le nombre de lignes de l'ecran */

/*-----------------------------------------------------------*/

short nbcolonne_ecran( void )

/*  retourne le nombre de colonnes de l'ecran */

/*-----------------------------------------------------------*/

void complete(char *t, char c)

/*  complete t_f par la touche fonction de sequence t et de code c
    si t est non vide ,sinon on ne fait rien */

/*-----------------------------------------------------------------*/

/*  procedure d'interception de signaux

ATTENTION :   ne sert plus a rien

void cntl_signx(SIG_PF numero)

            */

/*-----------------------------------------------------*/

void effac_lg(short l)

/* efface la ligne l
   ATTENTION :  0 < l <= LINES          */

/*-----------------------------------------------------*/

void weffac_lg(WINDOW *win, short l)

/* efface la ligne l dans la fenetre win
   ATTENTION :  0 < l <= LINES          */

/*-----------------------------------------------------*/

void ins_lg(short l)
```

```
/* insere a l'ecran une ligne en position (l,1)  , 0 < l <= LINES */

/*-------------------------------------------------------*/

void wins_lg(WINDOW *win, short l)

/* insere dans win une ligne en position (l,1)  , 0 < l <= LINES */

/*-------------------------------------------------------*/

void sup_lg(short l)

/*  supprime a l'ecran la ligne de rang l  0 < l <= LINES */

/*-------------------------------------------------------*/

void wsup_lg(WINDOW *win,short l)

/*  supprime dans win la ligne de rang l  0 < l <= LINES */

/*-------------------------------------------------------*/

void effac_rect(short l,short c,short h,short lg)

/*
     efface le rectangle d'origine superieure
     droite (l,c) de hauteur h <= LINES et de largeur lg <= COLS
     0 < l <= LINES et  0 < c <= COLS
*/

/*-------------------------------------------------------*/

void weffac_rect(WINDOW *win,short l,short c,short h,short lg)

/*
     efface dans win le rectangle d'origine superieure
     droite (l,c) de hauteur h <= LINES et de largeur lg <= COLS
     0 < l <= LINES et  0 < c <= COLS
*/

/*-------------------------------------------------------*/

void rect(short l,short c,short h,short lg)

/*
   trace le rectangle d'origine superieure droite (l,c)
   de hauteur h <= LINES et de largeur lg <= COLS
     0 < l <= LINES et  0 < c <= COLS
*/

/*-------------------------------------------------------*/

void wrect(WINDOW *win ,short l,short c,short h,short lg)

/*
   trace dans win le rectangle d'origine superieure droite (l,c)
   de hauteur h <= LINES et de largeur lg <= COLS
     0 < l <= LINES et  0 < c <= COLS
*/
```

```
/*-------------------------------------------------------*/

void woption(WINDOW *win,char *table)

/*
active dans win les attributs video codes sous forme de chaine
de caracteres dans table
si un attribut est inexistant alors on n'en tient pas
compte.
*/

/*-------------------------------------------------------*/

void option(char *table)

/*
active les attributs video codes sous forme de chaine
de caracteres dans table
si un attribut est inexistant alors on n'en tient pas
compte.
*/

/*-------------------------------------------------------*/

void f_option(void)

/*
desactive les attributs de visualisation
et retour a la normale   */

/*-------------------------------------------------------*/

void wf_option(WINDOW *win)

/*
desactive dans win les attributs de visualisation
et retour a la normale   */

/*-------------------------------------------------------*/

void option_def(char *table)

/*
active l'attribut cache, s'il existe et si table est non
vide ,sinon active la sequence d'attributs quelconques
definie par table.
option_def est utilise uniquement par lire_string.
*/

/*-------------------------------------------------------*/

void woption_def(WINDOW *win,char *table)

/*
active dans win l'attribut cache, s'il existe et si table est non
vide ,sinon active la sequence d'attributs quelconques
definie par table.
option_def est utilise uniquement par lire_string.
*/

/*-------------------------------------------------------*/
```

```
void f_option_def(void)

/*
desactive l'attibut cache s'il existe, sinon desactive
la sequence d'attributs actifs.

*/

/*-------------------------------------------------------*/

void wf_option_def(WINDOW *win)

/*
desactive dans win l'attibut cache s'il existe, sinon desactive
la sequence d'attributs actifs.

*/

/*-------------------------------------------------------*/

void xy(short l,short c)

/*
positionne le curseur en ligne l et colonne c .

REMARQUE :
     pour l'utilisateur les Nos de lignes vont de 1 a LINES
                 les Nos de colonnes vont de 1 a COLS

     en interne , les nos de lignes vont de 0 a LINES -1 et
              les Nos de colonnes vont de 0 a COLS -1;

*/

/*-------------------------------------------------------*/

void wxy(WINDOW *win,short l,short c)

/*
positionne dans win le curseur en ligne l et colonne c .

REMARQUE :
     pour l'utilisateur les Nos de lignes vont de 1 a LINES
                 les Nos de colonnes vont de 1 a COLS

     en interne , les nos de lignes vont de 0 a LINES -1 et
              les Nos de colonnes vont de 0 a COLS -1;

*/

/*-------------------------------------------------------*/

void effac(void )

/*  effacement ecran */

/*-------------------------------------------------------*/

void weffac(WINDOW *win)
```

```
/*  effacement de win */

/*-------------------------------------------------------*/

void effac_part(short l)

/*
   effacement partiel de l'ecran a partir de la ligne l incluse
         0 < l <= LINES
*/

/*-------------------------------------------------------*/

void weffac_part(WINDOW *win,short l)

/*
   effacement partiel de win  a partir de la ligne l incluse
         0 < l <= LINES
*/

/*-------------------------------------------------------*/

void bell(void)

/*  sonnette  */



/*-------------------------------------------------------*/

void ligne_h(short l,short c,short n)

/*
     trace une  ligne horizontale en (l,c) de longueur n
         0 < l <= LINES et 0 < c <= COLS
*/

/*-------------------------------------------------------*/

void wligne_h(WINDOW *win,short l,short c,short n)

/*
     trace dans win une  ligne horizontale en (l,c) de longueur n
         0 < l <= LINES et 0 < c <= COLS
*/

/*-------------------------------------------------------*/

void ligne_v(short l,short c,short n)

/*
   ligne verticale en (l,c) descendante de longueur n
        0 < l <= LINES et 0 < c <= COLS
 */

/*-------------------------------------------------------*/

void wligne_v(WINDOW *win,short l,short c,short n)

/*
   trace dans win une ligne verticale en (l,c) descendante de
```

```
   longueur n
        0 < l <= LINES et 0 < c <= COLS
 */

/*-------------------------------------------------------*/

void init_term(void )

/*
   initialise le tableau t_f avec les codes des touches fonctions
   et des touches de deplacement
   remarque :
      si la variable systeme VARTF est positionnee, alors c'est elle
      qui permet d'attribuer la valeur interne a chaque touche fonction
      ou deplacement, sinon on utlise les valeurs par defaut.

        la variable systeme SEQTF permet , si elle est positionnee, de
        preciser des sequences particulieres associees a certaines
        touches fonctions ( utile quand il y a des pbs de configuration
        de terminaux.

        exemple :

la sequence suivante sur Wyse 75:

           SEQTF="6 A 7 C 1 1"

permet d'associer a la touche fonction de code interne A  la sequence ^F
 (valeur generee = 6), a la touche fonction de code interne C la sequence ^G
(valeur 7)  et a fleche haut la sequence ^A (valeur 1)

ATTENTION :

  - Ces sequences supplementaires ne remplacent pas les sequences
    standards mais elles les completent en donnant une autre solution possible.

  - Si vous positionnez en meme temps SEQTF et VARTF, il vous appartient
    de verifier la coherence du code interne des touches fonctions

  - Sur les terminaux X vous pouvez utiliser les sequences ALT...
    par exemple ALT1 ALT2, ALT3,etc  associees aux touches F1 F2 F3

*/

/*-------------------------------------------------------*/

void ecri(short l,short c,char *s,char *topt)

/* ecrit la chaine s en (l,c) avec option topt  */

/*-------------------------------------------------------*/

void wecri(WINDOW *win,short l,short c,char *s,char *topt)

/* ecrit dans win la chaine s en (l,c) avec option topt  */

/*-------------------------------------------------------*/

void ecri_ent(short l,short c,short e,char *topt)

/* ecrit un entier e en l,c ou rien si e=ent_vid
```

```
   avec les attributs de visualisation de topt
   e : de type short
*/


/*-------------------------------------------------------*/

void wecri_ent(WINDOW *win,short l,short c,short e,char *topt)

/* ecrit dans win un entier e en l,c ou rien si e=ent_vid
   avec les attributs de visualisation de topt
   e : de type short
*/


/*-------------------------------------------------------*/

void ecri_lgent(short l,short c,long e,char *topt)

/* ecrit un entier e en l,c ou rien si e=ent_vid
   avec les attributs de visualisation de topt
   e : de type long
*/


/*-------------------------------------------------------*/

void wecri_lgent(WINDOW *win,short l,short c,long e,char *topt)

/* ecrit dans win un entier e en l,c ou rien si e=ent_vid
   avec les attributs de visualisation de topt
   e : de type long
*/


/*-------------------------------------------------------*/

void ecri_lgmnt(short l,short c,double f,char *topt)

/* ecrit un montant long f en l,c ou rien si f = vide
   avec les attributs de visualisation de topt */

/*-------------------------------------------------------*/

void wecri_lgmnt(WINDOW *win,short l,short c,double f,char *topt)

/* ecrit dans win un montant long f en l,c ou rien si f = vide
   avec les attributs de visualisation de topt */

/*-------------------------------------------------------*/

char code(char c,short plage)

/* renvoie un code associe au caractere c suivant les
   conventions :

   code retour       caractere

      1           CR
      2           BS
      3           DEL
```

```
      4           non accepte
      5           touche fonction ou fleche deplacement ou
                  autre touche speciale
      6           caractere accepte defini par plage


   plage definit les caracteres acceptes suivant la convention:

        0               aucun caractere
        1               chaine de caracteres quelconques
            2               chiffres
            3               chiffres et '/', ' ', '-'
        4               chaine de caracteres en majuscules
        5               chiffres et '+' , '-'
            6           chiffres et '+' , '-' , '.'

*/

/*-------------------------------------------------------*/

static short ch_inch_c(char *s1,char *s2)

/* recherche si la chaine s1 apparait exactement suivi d'un espace
   dans la chaine s2.
   si oui, retourne le 1er caractere non blanc de s2 suivant l'apparition
   de s1.
   sinon retourne le caractere '!'
   cette fonction est utilisee par lire_string dans la version curses
*/

/*-------------------------------------------------------*/

char ch_inch(char *s1,char *s2)

/*
   determine si la chaine s1 est partiellement ,totalement
   ou n'est pas incluse dans la chaine s2.
   s2 peut contenir un nombre quelconque de sous-chaines
   separees par un seul espace y compris avant EOS.
   la procedure renvoie un caractere egal a :

      - ! si s1 n'apparait pas dans s2
      - ? si s1 apparait partiellement dans s2 (en debut de sequence)
      - 1er caractere non blanc de s2 suivant
        l'apparition de s1 dans le cas ou s1
      est exactement egale a une sous-chaine de s2.
*/



/*-------------------------------------------------------*/

short lire_string(short l,short c,char *s,short n,char *topt,
             char *valid,short plage)

/*
   lecture en (l,c) d'une chaine de caracteres s de
   longueur n. si s est non vide a l'appel de la procedure
   il y a affichage de s.

   le parametre "plage" definit les caracteres acceptes
```

```
   par la procedure (cf. code(c,plage)).

   les regles de saisie sont les suivantes:

   si le 1er caractere saisi est soit return,soit une fleche
   de deplacement soit une touche fonction alors fin
   de saisie et s inchangee,sinon s est effacee et on
   commence une nouvelle saisie. la fin de saisie est alors
   obtenue soit par l'une des touches precedentes ou par
   le nombre de caracteres n atteint.
   il est possible de modifier eventuellement une partie de la chaine
   s sans pour autant tout effacer . C'est le mode CORRECTION
   active/desactive par la touche de fonction predefinie CORREC.

   dans le mode CORRECTION

      - "fleche droite","fleche gauche" permettent
        de se deplacer a droite/gauche dans la chaine s,

      - la touche BS (back space) permet d'effacer le caractere pointe
        par le curseur (si on est dans la chaine s) ou le caractere precedant
          (si on est apres le dernier caractere de s).

   NB : on peut passer dans le mode CORRECTION en cours de saisie

   le parametre "opt" decrit l'ensemble des  attributs video

   le parametre "valid" decrit l'ensemble des touches autorisees
   provoquant la fin de saisie, codees sous la forme d'un caractere :

               - 1 : fleche haut
               - 2 : fleche droit
               - 3 : fleche bas
               - 4 : fleche gauche
               - A : touche fonction 1
               _ B : touche fonction 2
                 .
                 .
               - Z : touche fonction 26

  return est considere comme fin de saisie standard.

  la procedure lire renvoie un code retour entier defini
  de la facon suivante :

    octet de poids fort   ---->  1 si la chaine s a ete modifiee

                                 0 sinon

    octet de poids faible ---->  0 si fin de saisie par return ou n atteint

                              1,2,..,Z suivant la touche fin de saisie.

*/

/*-----------------------------------------------------*/

short wlire_string(WINDOW *win,short l,short c,char *s,short n,
               char *topt,char *valid,short plage)

/*
```

```
        idem a lire_string mais la saisie se fait dans win
*/

/*-------------------------------------------------------------------------*/

short lire_curs(short l,short c,char *cara)

/*
      Attend la frappe d'une touche a la position l,c .
      Aucun affichage de caractere n'est realise .
        Le curseur n'est pas deplace.

      Code retour (en binaire) :

            - 0 la touche enfonce est une fleche
            - 1                 une touche de fonction
            - 2                 un caractere affichable
            - 3                 CR
            - 4                 BS
            - 5                 un caractere non affichable

      le caractere contient respectivement le code de la fleche ,
      le code de la fonction , le caractere affichable , CR , BS ,
      le caractere non affichable .
*/

/*-------------------------------------------------------------------------*/

short wlire_curs(WINDOW *win,short l,short c,char *cara)

/*

      idem que pour lire_curs mais dans le fenetre win

*/

/*----------------------- FIN DE FICHIER -------------------------------*/

/*



          FICHIER PRIMITIVES DE BASE (2eme partie)
          ----------------------------------------

ATTENTION : ce fichier doit etre compile avec l'option -DV7 si l'on
          est en unix version 7
                              avec l'option -DCBREAK si l'on
          veut le mode cbreak plutot que le mode raw.

                  version curses.h

                    ( multi-fenetres )

                      ( O.B.J.E.T )

*/

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

```
#include <ctype.h>
#include <curses.h>
#include "pconstante.h"

#include <errno.h>
#include "prototype.h"

/* double atof (char *) ;   */

/*-------------------------------------------------------*/

short tst_ch_ent(char *s)

/*  fonction qui teste si une chaine s est
    numerique
    renvoie   : 1 si s est numerique
              : 0 sinon
*/

/*-------------------------------------------------------*/

void arrondir(double *f)

/* arrondit le montant long au centime inferieur ou superieur
   doit etre appele avant tout affichage et stockage d un montant
   long dans un fichier */

/*-------------------------------------------------------*/

short tst_lgmnt_vid(double f)

/* renvoie 1 si le montant long est vide ie = LGMNT_VIDE, 0 sinon */

/*-------------------------------------------------------*/

void set_lgmnt_vid(double *f)

/* donne a f la valeur vide = LGMNT_VIDE */

/*-------------------------------------------------------*/

short tst_lgent_vid(long e)

/* renvoie 1 si entier long vide ie = MAX_VAL_LG  0 sinon  */

/*-------------------------------------------------------*/

void set_lgent_vid(long *e)

/* donne a e la valeur vide = MAX_VAL_LG  */

/*-------------------------------------------------------*/

short tst_ent_vid(short e)

/* renvoie 1 si entier court vide ie =MAX_VAL_CRT  0 sinon  */

/*-------------------------------------------------------*/

void set_ent_vid(short *e)
```

```
/* donne a l'entier court e la valeur vide = MAX_VAL_CRT */

/*-------------------------------------------------------*/

void set_ch_vid(char *s)

/*  donne a  la chaine s la valeur vide */

/*-------------------------------------------------------*/

short tst_ch_vid(char *s)

/*   renvoie 1 si s est vide (*s egal '\0')
            0 sinon   */

/*-----------------------------------------------------------------*/

short egal_ch(char *s1,char *s2)

/* code retour = 1 si les 2 chaines ont meme dimension et sont egales
               ou si la plus grande ne differe que par des blancs en
               fin de chaine par rapport a la plus petite chaine;
               ex: 'abc' et 'abc   ' sont consideres comme egales.
               = 0 sinon.
*/

/*-----------------------------------------------------------------*/

void copy(char *s1,char *s2,short lgs2)

/*
   copie lgs2 caracteres de s1 vers s2 et positionne dans tous les cas
   EOS en lgs2+1 dans s2.
   si la longueur de s1 est inferieure a lgs2 on complete s2 par ' ',
   sinon on tronque
*/

/*-----------------------------------------------------------------*/

char *fin_chaine(char *s)

/*
  met \n\0 apres le dernier caractere non blanc de s.
  on suppose qu'il y a sufisamment de place et que l'on trouve \0 .
  la fonction retourne le pointeur sur \n .
*/

/*-------------------------------------------------------------*/

void cpl_ch(char *s,short lg)

/* complete  a lg caracteres la chaine s en ajoutant
   des blancs . il doit rester assez de place dans s */

/*-------------------------------------------------------------*/

short inclus(char *s1,char *s2)

/*
renvoie le rang de debut de s1 dans s2 si la chaine
s1 est incluse dans s2
```

```
renvoie  0 sinon
attention : le rang commence a 1 .  */

/*-------------------------------------------------------*/

char car_inch(char c,char *s)

/*
renvoie un entier > 0 si le caractere c apparait
               dans la chaine s
          = 0 sinon
*/



/*-------------------------------------------------------*/

void ecr_err(char *mess)

/*
affiche en (LG_ERR,1) le message mess,puis lit
return et efface le message
*/

/*-------------------------------------------------------*/

void wecr_err(WINDOW *win,char *mess)

/*
     affiche dans win , en avant derniere ligne de win le message mess,puis lit
     return et efface le message
ATTENTION : on suppose que le bord de la fenetre est trace, ce qui oblige
     a afficher le message sur l'avant derniere ligne et a partir de la position
     2
*/

/*-------------------------------------------------------*/

void erreur(char *mess,short n)

/*
   affiche le message d'erreur mess ,ainsi que le
   numero de l'erreur. et rend la main au systeme par un  exit().
*/

/*-------------------------------------------------------*/

void werreur(WINDOW *win,char *mess,short n)

/*
   affiche dans win le message mess ,ainsi que le
   numero de l'erreur. et rend la main au systeme par un  exit().
*/

/*-------------------------------------------------------*/

void mode_raw(void )

/*-------------------------------------------------------*/

void mode_cooked(void )
```

```
/*-------------------------------------------------------*/

void deb_vac(void )

/* a execute avant toute utilisation des outils
   suppression du l echo et passe en mode raw ou cbreak */

/*-------------------------------------------------------------*/

void fin_vac(void )

/* fin d'une vacation : restaure echo et mode normal */

/*-------------------------------------------------------------*/

void ecri_lgent_l(short l,short c,long e,short n,char *opt)

/*
   ecrit l'entier e de type long en (l,c) avec les attributs opt
   si e est non vide et avec cadrage a droite par rapport a la
   zone d'affichage de longueur n.
*/

/*-------------------------------------------------------------*/

void wecri_lgent_l(WINDOW *win,short l,short c,long e,short n,char *opt)

/*
   ecrit dans win l'entier e de type long en (l,c) avec les attributs opt
   si e est non vide et avec cadrage a droite par rapport a la
   zone d'affichage de longueur n.
*/

/*-------------------------------------------------------*/

void ecri_ent_l(short l,short c,short e,short n,char *opt)

/* ecrit un entier court e en l,c ou rien si entier vide, avec les
   attributs opt .
   il y a cadrage a droite par rapport a la zone d'affichage de
   longueur n.
*/

/*-------------------------------------------------------*/

void wecri_ent_l(WINDOW *win,short l,short c,short e,short n,char *opt)

/*
      idem a ecri_ent_l mais dans win
*/

/*-------------------------------------------------------*/

void ecri_l(short l,short c,char *s,short n,char *opt)

/*
   ecrit la chaine s en l,c avec les attributs video opt.
   complete ensuite a l'ecran par des blancs jusqu'a n
   caracteres
*/
```

```
/*-------------------------------------------------------*/

void wecri_l(WINDOW *win,short l,short c,char *s,short n,char *opt)

/*
      idem a ecri_l mais dans win
*/

/*-------------------------------------------------------*/

short lire_ch(short l,short c,char *s,short n,char *topt,char *valid)

/* lecture d'une chaine de caracteres s de longueur n
   la  procedure suit les regles definies pour lire_string.
*/

/*-------------------------------------------------------*/

short wlire_ch(WINDOW *win,short l,short c,char *s,short n,
            char *topt,char *valid)

/* lecture dans win d'une chaine de caracteres s de longueur n
   la  procedure suit les regles definies pour lire_string.
*/

/*-------------------------------------------------------*/

short lire_chq(short l,short c,char *s,short n,char *topt,char *valid)

/* lecture d'une chaine s de caracteres quelconque et de longueur n
   la  procedure suit les regles definies pour lire_string.
*/

/*-------------------------------------------------------*/

short wlire_chq(WINDOW *win,short l,short c,char *s,short n,
            char *topt,char *valid)

/* lecture dans win d'une chaine s de caracteres quelconque et de longueur n
   la  procedure suit les regles definies pour lire_string.
*/

/*-------------------------------------------------------*/

short lire_ent(short l,short c,short *e,short n,char *topt,char *valid)

/* lecture en (l,c) d'un entier relatif e < MAX_VAL_CRT
   (en valeur absolue) et de longueur n <= 6.
   la procedure suit les regles definies pour lire_string.
*/

/*-------------------------------------------------------*/

short wlire_ent(WINDOW *win,short l,short c,short *e,short n,
            char *topt,char *valid)

/* lecture dans win en (l,c) d'un entier relatif e < MAX_VAL_CRT
   (en valeur absolue) et de longueur n <= 6.
   la procedure suit les regles definies pour lire_string.
*/
```

```
/*-------------------------------------------------------*/

short lire_lgent(short l,short c,long *e,short n,
               char *topt,char *valid)

/* lecture en (l,c) d'un entier relatif e < MAX_VAL_LG (en valeur
   absolue) et de longueur n <= 10 (avec 9 chiffres significatifs au max) .
   la procedure suit les regles definies pour lire_string.
*/

/*-------------------------------------------------------*/

short wlire_lgent(WINDOW *win,short l,short c,long *e,
              short n,char *topt,char *valid)

/*
      idem a lire_lgent mais dans win
*/

/*-------------------------------------------------------------*/

void ecri_lgmnt_l(short l,short c,double f,short n,char *opt)

/* idem a ecri_lgmnt avec cadrage a droite sur la zone d affichage de
   longueur n qui comprend tjs le . et les 2 chiffres decimaux (n>=4) */

/*-------------------------------------------------------------*/

void wecri_lgmnt_l(WINDOW *win,short l,short c,double f,short n,char *opt)

/* idem a ecri_lgmnt_l mais dans win */

/*-------------------------------------------------------------*/

short lire_lgmnt(short l,short c,double *f,short n,
             char *topt,char *valid)

/*
  lecture en (l,c) d'un montant long f ,sur une zone de longueur
  totale n >= 4 et n <= 14.
  la saisie du point decimal et des 2 chiffres decimaux n'est pas
  obligatoire, mais ils apparaissent toujours apres validation de la
  saisie. ( maximun 10 chiffres significatifs avant le point)
*/

/*-------------------------------------------------------------*/

short wlire_lgmnt(WINDOW *win,short l,short c,double *f,short n,
              char *topt,char *valid)

/*
      idem a lire_lgmnt mais dans win
*/

/*-------------------------------------------------------------*/


/*---------------------- FIN DU FICHIER -----------------------*/

/*
```

```
                    FICHIER DES PRIMITIVES DE BASE (3eme PARTIE)
                    --------------------------------------------


                   version curses.h

                    multi-fenetres

                      ( O.B.J.E.T )

*/

#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <curses.h>
#include "pconstante.h"

#include <errno.h>
#include "prototype.h"

/* extern "C" long int atol(const char *);

  a mettre pour le conpilateur C++  */

/*-------------------------------------------------------------------------*/

void set_mnt_vid(long *e)

/*    donne a e la valeur vide = MAX_VAL_LG
*/



/*-------------------------------------------------------------------------*/

short tst_mnt_vid(long e)

/*    renvoie 1 si montant court vide e==MAX_VAL_LG
*/

/*-------------------------------------------------------------------------*/

void ecri_mnt(short l,short c,long e,char *topt)

/*    ecrit un montant e en l,c ou rien si e=amt_vid
      avec les attributs de visualisation de topt
      e : de type long
*/

/*-------------------------------------------------------------------------*/

void wecri_mnt(WINDOW *win,short l,short c,long e,char *topt)

/*
      idem a ecri_mnt mais dans win
*/

/*-------------------------------------------------------------------------*/
```

```
void ecri_mnt_l(short l,short c,long e,short n,char *opt)

/*    ecrit un montant court e en l,c ou rien  si montant vide, avec les
      attributs opt.
      il y a cadrage a droite par rapport a la zone d'affichage de longeur
      n.
*/

/*-------------------------------------------------------------------------*/

void wecri_mnt_l(WINDOW *win,short l,short c,long e,short n,char *opt)

/*
      idem a ecri_mnt_l mais dans win
*/

/*-------------------------------------------------------------------------*/

short lire_mnt(short l,short c,long *e,short n,char *topt,char *valid)

/*    lecture en l,c d'un montant e < MAX_VAL_LG ( en valeur absolue )
      et de longueur n <= 10 ( avec 9 chiffre significatifs au max ).
      la procedure suit les regles definies pour lire_string.
*/

/*-------------------------------------------------------------------------*/

short wlire_mnt(WINDOW *win,short l,short c,long *e,short n,
            char *topt,char *valid)

/*
      idem a lire_mnt mais dans win
*/

/*-------------------------------------------------------------------------*/

char lire_rep(short l,short c,char *s,short lg)

/* lit un caractere en l,c sur lg positions (lg = 1 ou lg =2), verifie
   que le caractere lu appartient a l'ensemble des caracteres possibles
   decrits par s, sinon redemande la lecture.
   (si lg = 2 alors le 2eme caractere tape doit etre return )

   code retour = caractere lu
*/

/*-------------------------------------------------------------------------*/

char wlire_rep(WINDOW *win,short l,short c,char *s,short lg)

/*
      idem a lire_rep mais dans win
*/

/*-------------------------------------------------------------------------*/

void barre_menu(short lg,char **menu,short sel)

/*
   affiche en ligne lg la barre menu dont les elements sont
```

```
   dans le tableau de pointeurs de car menu (dernier elt = pointeur nul)
   en mettant en forte intensite le choix de No sel ;
*/

/*-------------------------------------------------------------------------*/

void wbarre_menu(WINDOW *win,short lg,char **menu,short sel)

/*
      idem a barre_menu mais dans win
*/

/*------------------------------ FIN FICHIER ------------------------------*/

/*



          FICHIER PRIMITIVES DE BASE (4eme partie)
          ----------------------------------------

                    (version curses.h )

                   multi-fenetrage

                  (  O.B.J.E.T )

*/

#include <stdio.h>
#include<stdlib.h>
#include <signal.h>
#include <ctype.h>
#include <curses.h>
#include <string.h>
#include "pconstante.h"
#include <errno.h>

#include "prototype.h"

// fichier des primitives d'affichage integrant la couleur

/*-----------------------------------------------------*/

void wecri_c(WINDOW *win,short l,short c,char *s,int couleur, int fond)

/* ecrit dans win la chaine s en (l,c) avec couleur donnee   */

/*-----------------------------------------------------*/

void ecri_ent_c(short l,short c,short e,int couleur, int fond)

/* ecrit un entier e en l,c ou rien si e=ent_vid
   avec la couleur donnee en parametre   On remet en place la couleur du fond
      apres  affichage
   e : de type short
      il faut que le systeme de couleur soit mis en place
*/


/*-----------------------------------------------------*/
```

```
void wecri_ent_c(WINDOW *win,short l,short c,short e,int couleur, int fond)

/* ecrit dans win un entier e en l,c ou rien si e=ent_vid
   avec la couleur  donnee
   e : de type short
*/


/*-------------------------------------------------------*/
void wecri_ent_l_c(WINDOW *win,short l,short c,short e,short n,int couleur, int
fond)

/*
      idem a ecri_ent_l mais dans win
*/

/*-------------------------------------------------------*/
```