INRA-ESR

78850 Grignon

ORSTOM
213, rue La Fayette
75480 Paris cedex 10

ENVIRONNEMENT D'INTERFACE UTILISATEUR POUR LA CONCEPTION DE LOGICIELS D'AIDE A LA DECISION EN AGRICULTURE

J.C. POUSSIN, Chargé de Recherche ORSTOM

ENVIRONNEMENT D'INTERFACE UTILISATEUR POUR LA CONCEPTION DE LOGICIELS D'AIDE A LA DECISION EN AGRICULTURE

J.C. POUSSIN, Chargé de Recherche ORSTOM

oooooooooooooo

Les travaux de l'équipe dirigée par J.M. ATTONATY au laboratoire d'Economie et de Sociologie Rurale de l'INRA à Grignon, sont focalisés sur la modélisation des pratiques de gestion dans l'entreprise agricole. Avec le développement de la micro-informatique, ces travaux ont conduit à la conception de logiciels informatiques permettant d'aider les agriculteurs dans leurs décisions.

Afin d'être largement utilisés, ces outils destinés à de non spécialistes, doivent être faciles d'emploi, tant au niveau de l'utilisation des résultats (temps de traitement, lisibilité des résultats), qu'à celui de l'entrée des données.

Ayant décider d'utiliser un langage de programmation en mode compilé pour accélérer les traitements (langage C), nous avons constitué une bibliothèque d'outils de programmation (définitions et fonctions) permettant la conception et la gestion de l'interface utilisateur dans les logiciels concus au laboratoire. Toutes les personnes travaillant dans l'équipe ont participé à son élaboration et sa maintenance.

On dénombre actuellement environ 200 fonctions réparties en 7 familles :

- les fonctions de gestion de l'écran (tabulation, couleurs et fenêtrage),
- les fonctions d'impression,
- les fonctions de gestion de menus déroulants,
- les fonctions d'aide (mode d'emploi intégré),
- les fonctions de recherche de fichier ou de répertoire,
- les fonctions d'entrée et affichage formatés,
- les fonctions de saisie d'une fiche,
- les fonctions de gestion de "picklists" (choix d'un terme dans une liste).

La documentation, réalisée parallèlement à la programmation de ces outils, comprend :

- une fiche descriptive pour chaque famille (problématique, marche à suivre, définitions utilisées),
- une fiche descriptive pour chaque fonction (arguments, valeur de retour, mise en oeuvre, fonctionnement),
- des exemples de programmes utilisant ces fonctions.

Les trois dernières familles ont été élaborées plus spécialement par J.C. POUSSIN. Les pages ci-après sont extraites de la documentation.

L'entrée d'une donnée se fait grâce aux fonctions de la famille E_ENTDON() & C°. Il existe des **fonctions générales** pour la saisie de n'importe quel type de donnée, mais la librairie contient également les **fonctions adaptées** à la saisie d'un entier (int, unsigned, long), d'un réel (float, double), d'un caractère, d'une chaîne, d'un texte, d'une date, et même d'un nom de fichier ou de répertoire. Dans la plupart des cas, vous n'utiliserez que ces fonctions spécialisées.

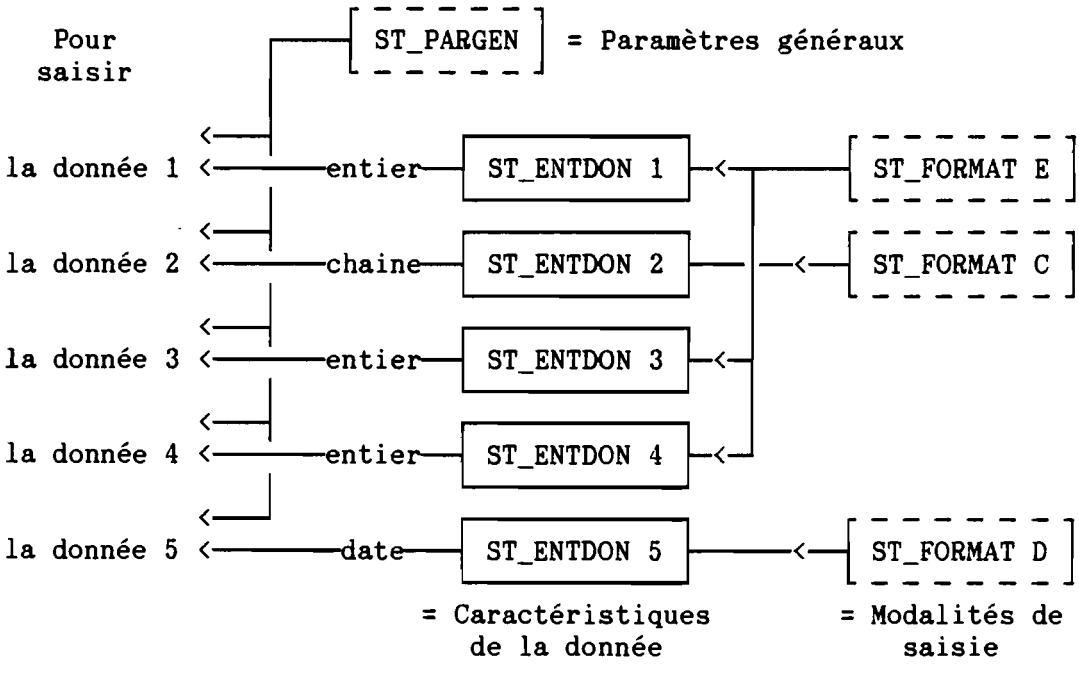
Ces fonctions font appel à trois ensembles de paramètres:

- des paramètres généraux, indépendants du type de donnée saisie et communs à un écran de saisie ou même toute une application, regroupés dans une structure ST_PARGEN;
- des paramètres caractérisant la donnée (notamment son adresse) et la zone de saisie (n° d'écran ou de palette, coordonnées ...);
- des paramètres décrivant les modalités de saisie d'un type de zone (conversion de la donnée, caractères valides, vérification ...), regroupés dans une structure ST_FORMAT.

Les deux derniers ensembles de paramètres sont regroupés dans une structure ST_ENTDON. Celle-ci est propre à la donnée à saisir, et contient une structure ST_FORMAT qui peut être communes à plusieurs données à saisir, et donc à plusieurs structures ST_ENTDON.

Les fonctions spécialisées utilisent une structure ST_PARGEN mais créent automatiquement la structure ST_ENTDON adéquate.

Toutes ces fonctions fonctionnent en écran physique et en écrans virtuels (VSI).



1. Marche à suivre

0 - Facultatif : définir les paramètres généraux de la saisie

Déclarer une structure ST_PARGEN, et l'initialiser grâce à e_pargen()

(Cf. Remarque du paragraphe 2.1)

A - Une donnée à saisir : choisir la fonction e_ent???() adéquate

- e_entcar() : un caractère
- e_entalp() : une chaîne
- e_enttxt() : un texte
- e_entdat() : une date
- e_entfic() : un nom de fichier
- e_entint() : un entier de type int
- e_entuns() : un entier de type unsigned
- e_entlng() : un entier de type long
- e_entflt() : un réel de type float
- e_entdbl() : un réel de type double

Toutes ces fonctions attendent comme arguments l'adresse de la donnée à saisir, l'adresse d'un drapeau de modification (au retour il est égal à OUI si la donnée a été modifiée); elle retournent le code de la touche de sortie.

IMPORTANT : Les structures ST_ENTDON et ST_FORMAT associées à la donnée saisie sont créées automatiquement.

B - Plusieurs données à saisir et gestion direct par l'utilisateur :

Utilisation de la fonction générale **e_entdon()** qui a pour arguments l'adresse de la structure ST_PARGEN, et l'adresse d'une structure ST_ENTDON associée à la donnée à saisir.
En conséquence la marche à suivre est la suivante :

1) Les diverses déclarations :

. *Obligatoires* :

Déclarer une structure ST_ENTDON pour chaque donnée à saisir

. *Facultatives* : (Cf. Remarque du paragraphe suivant)

Déclarer une structure ST_FORMAT pour chaque type de modalités de saisie.
Une structure ST_FORMAT peut être commune à plusieurs données (donc à plusieurs structures ST_ENTDON) à condition qu'elles soient de même type (int, char*, ...).

2) L'initialisation des structures :

. Initialiser les structures ST_FORMAT déclarées :

* soit en utilisant les fonctions adaptées à chaque type de zone e_fmtcar(), e_fmtalp(), e_fmtint(), e_fmtuns(), e_fmtlng(), e_fmtflt(), e_fmtdat()

* soit par e_fmtdon(), qui permet de personnaliser entièrement les modalités de saisie

. Initialiser les structures ST_ENTDON à l'aide des fonctions e_zon???() (une pour chaque type de donnée)

Remarques : Ces fonctions ont toutes en argument l'adresse de la structure ST_FORMAT associée à la zone de saisie. Si ce pointeur est NULL, la fonction utilise un 'format' standard, propre au type de la donnée.

3) La saisie proprement dite :

Elle se fait grâce à e_entdon(). Cette fonction attend pour arguments l'adresse de la structure ST_PARGEN utilisée (ou NULL), et celle de la structure ST_ENTDON décrivant la donnée à saisir.

Elle retourne le code de la touche de sortie; un drapeau placé dans la structure ST_ENTDON permet de savoir si la donnée a été modifiée.

C - Plusieurs données à saisir et gestion automatique :

On distingue deux grands types :

- la saisie de type FICHE
  (se référer au chapitre **SAISIE D'UNE FICHE**)

- la saisie de type TABLEAU
  (se référer au chapitre **SAISIE D'UN TABLEAU**)

D - Afficher une donnée saisie :

e_affdon() - attend comme argument la structure ST_ENTDON associée à la donnée à afficher.

2. Les diverses structures utilisées

2.1 Les paramètres généraux de saisie

Ces paramètres définissent des modalités de saisie qui sont communes à un écran de saisie ou même à une application et non pas particuliers à une donnée ou une zone de saisie. Il sont contenus dans une structure **ST_PARGEN.**

```
ST_PARGEN
        {

        int Sortieg;        /* = OUI ou NON : Autorisation de sortie de zone */
                            /* par  GAUCHE  ou  BACKSPACE  en début de zone  */

        int Sortied;        /* = OUI ou NON : Autorisation de sortie de zone */
                            /* par  DROIT  en fin de zone                    */

        int Depauto;        /* = OUI ou NON : Autorisation de sortie de zone */
                            /* quand la zone est complète                    */

        int Car_remp;       /* Caractère de remplissage de la fin de zone    */
                            /* en cours de modif                             */

        int T_mod;          /* Code de la touche d'entrée dans la zone ( ou 0 ) */
        int T_init;         /* Code de la touche de réinitialisation   ( ou 0 ) */
        int T_eff;          /* Code de la touche d'effacement de zone  ( ou 0 ) */

        } ;
```

Initialisation par e_pargen().

Utilisée par d_dicol(), e_saisie(), e_ent???(), et e_fentre().

Remarque : La définition d'une telle structure n'est pas obligatoire : l'envoi d'un pointeur NULL dans les fonctions e_ent???() déclenche l'utilisation d'une structure ST_PARGEN implicite (aucune sortie autorisée, remplissage par des BLANCS, aucune touche spéciale)

2.2 La description de la zone de saisie

ST_ENTDON décrit la zone de saisie d'une donnée à travers des paramètres concernant la donnée :

```
ST_ENTDON
        {
        int Ecran;          /* n° d'écran VSI, ou n° de palette
        int Lig, Col;       /* coordonnées du début de la zone de saisie
        int Lg;             /* longueur de la zone

        void *Donnee;       /* adresse de la donnée saisie

        ST_FORMAT *Format; /* adresse de la structure contenant les modalités
                           /* de saisie de la zone

        char *Chaine;       /* chaîne saisie

        int Ds_zone;        /* indicateur d'entrée dans la zone
        int No_ordre;       /* position du curseur dans la chaîne saisie
        int Modif;          /* drapeau de modification
        };
```

Initialisation par e_zon???()

Utilisée par d_dicol(), e_saisie(), e_ent???(), e_fentre(), e_affdon()

2.3 Les modalités de saisie d'une zone

ST_FORMAT décrit les modalités de saisie d'une zone associée à un certain type de donnée.

Ces modalités se traduisent par l'appel à des fonctions spécialisées :

- conversion de la donnée en chaine (Cvd : ConVersion Début)
- conversion de la chaine en donnée (Cvf : ConVersion Fin)
- vérification finale de la donnée (Ver : VERification)
- recadrage de la chaine pour affichage final (Cad : reCADrage)

] appelées par e_entdon()

- Filtre des Caractères Valides (Fcv)
- Filtres des Touches Spécifiques (Fts)

] appelées par e_fentre()

De plus cette structure contient un pointeur "boîte aux lettres" donnant accès à des renseignements complémentaires sur la donnée (Ptr)
(ex: pointeur sur une liste de caractères acceptables pour la saisie d'un caractère)

```
ST_FORMAT
       {
       char *(*Cvd) ( ST_ENTDON *ste );
       void  (*Cvf) ( ST_ENTDON *ste );          /* '*ste' est l'adresse
       int   (*Ver) ( ST_ENTDON *ste );          /* de la structure ST_ENTDON
       void  (*Cad) ( ST_ENTDON *ste );          /* associée à la donnée

       int   (*Fcv) ( int *s, ST_ENTDON *ste ); /* 's' est le code de
       int   (*Fts) ( int *s, ST_ENTDON *ste ); /* la touche frappée

       void *Ptr;
       } ;
```

Initialisation par e_fmt???()

2.4 Les structures de vérification d'un type de données

Ces structures contiennent les éléments qui permette de vérifier si la donnée saisie est conforme ou non. Leur adresse est mémorisée dans le pointeur "boîte aux lettres" de la structure ST_FORMAT.

* Vérification d'un entier

```
ST_VERINT
        {
        int Mini,   Maxi;          /* minimum et maximum raisonnable
        int Minabs, Maxabs;        /* minimum et maximum absolu
        char *Messmin, *Messmax;   /* messages de dépassement
                                   /* des bornes raisonnables
        } ;
```

Initialisée par e_fmtint(), e_fmtuns(), e_fmtlng()

* Vérification d'un réel

```
ST_VEREEL
        {
        int Nav, Nap;              /* nb de chifres avant et après la vigule
        double Mini,   Maxi;       /* minimum et maximum raisonnable
        double Minabs, Maxabs;     /* minimum et maximum absolu
        char *Messmin, *Messmax;    /* messages de dépassement
                                   /* des bornes raisonnables
        } ;
```

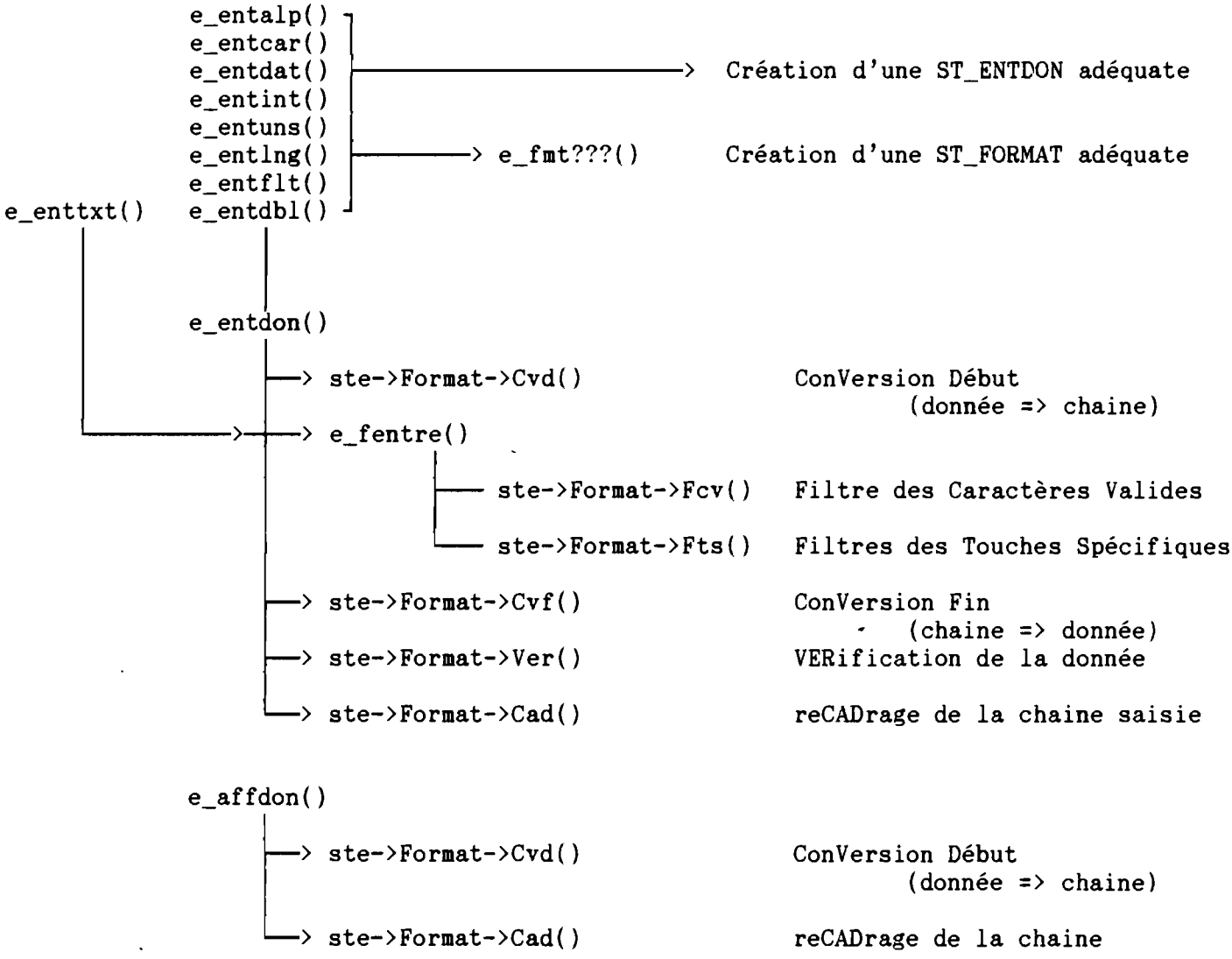
Initialisée par e_fmtflt(), e_fmtdbl()

* Vérification d'une date

```
ST_VERDAT
        {
        char *Format;              /* format d'une date (ex : "JJ/MM")
        char *Jour;                /* adresse du 1er 'J' dans le format
        char *Mois;                /* adresse du 1er 'M' dans le format
        char *An;                  /* adresse du 1er 'A' dans le format
        char *Mini;                /* date minimale absolue
        char *Maxi;                /* date maximale absolue
        };
```

Initialisée par e_fmtdat()

3. Organisation des fonctions



4. Précisions sur les fonctions de ST FORMAT

4.1 Les fonctions de la bibliothèque

Le nom des fonctions fournies dans la bibliothèque Grignon suit les règles suivantes :

- e_???_?()

- les 3 lettres du milieu représentent le type de fonction :
  - cvd : conversion début
  - cvf : conversion fin
  - ver : vérification
  - cad : recadrage
  - fcv : filtre des caractères valides
  - fts : filtre des touches spécifiques

- la dernière lettre représente le type de la variable, sauf pour fts qui ne dépend pas de ce type (pour les dates et les textes, 2 lettres)

  - a : chaîne alpha-numérique
  - c : caractère
  - dt: date
  - tx: texte
  - e : entier (int, unsigned, long)
  - i : int
  - u : unsigned
  - l : long
  - f : float
  - d : double

Exemple : e_ver_i() : fonction de vérification pour un entier int
e_cad_e() : fonction de recadrage pour un entier (int, unsigned ou long)

A priori, vous n'aurez donc pas à créer vos propres fonctions. Toutefois le filtre des touches spécifiques est typiquement à créer pour une application donnée (par exemple : choix d'un terme dans une "picklist", l'adresse de cette "picklist" étant mémorisée dans le pointeur "boîte aux lettres" de la structure ST_FORMAT) : vous pourrez alors modifier la fonction standard (e_fts_st()) en l'adaptant à vos besoins.

4.2 Les arguments de ces fonctions

Toutes ces fonctions attendent au moins en argument l'adresse de la structure ST_ENTDON en cours d'utilisation.
e_fcv_?() et e_fts_?() attendent en plus l'adresse du code de la touche pressée (il peut donc être modifié !)

4.2 Les valeurs de retour

- e_cvd_?() : l'adresse d'une chaîne
  (généralement allouée par calloc() ou malloc())

  ATTENTION : à la fin de e_entdon(), cette chaîne sera
  libérée par free() :
  if (chaine != donnee) free(chaine);

- e_cvf_?() et e_cad_?() : aucune

- e_ver_?() : retourne OUI si correct ou NON si incorrect

- e_fcv_?() : retourne CAR_OK ou INCORRECT ou A_TRAITER

- e_fts_?() : retourne DE_SORTIE ou TRAITE ou INCORRECT ou A_TRAITER

Valeurs de ces codes (à titre indicatif)

```
#define  CAR_OK      1
#define  TRAITE      1
#define  A_TRAITER   0          (Cf. grentre.h)
#define  INCORRECT  -1
#define  DE_SORTIE  -2
```

On se propose de saisir une "fiche", composée de plusieurs zones de saisie. Il faut donc décrire chacune des zones, puis gérer le déplacement d'une zone à l'autre. Ces caractéristiques, propres à chaque zone, sont rassemblées dans la structure ST_SAISIE; le plan de saisie de l'ensemble de la fiche sera alors un tableau de structure ST_SAISIE. La fonction e_saisie() suivra ce plan.

Dans le cas général, e_saisie() fera appel à e_entdon() pour la saisie d'une zone; mais il est possible de faire appel à sa propre fonction de saisie.

1. Marche à suivre

O - Facultatif : définir les paramètres généraux de la saisie

Déclarer une structure ST_PARGEN, et l'initialiser grâce à e_pargen()

B - Définir un plan de saisie :

Déclarer un tableau de structure ST_PARGEN

ST_PARGEN plan[<nb de zones à saisir>];

Puis,

1) Cas général : utilisation de e_entdon() pour la saisie de chaque zone

. Déclarer autant de structures ST_FORMAT qu'il est nécessaire et les définir (Cf. ENTREES FORMATEES)

2) Cas de l'utilisation de fonctions de saisie personnelles

. La structure ST_FORMAT associée est facultative
. La fonction dde saisie personnelle attend pour arguments l'adresse de la sturcture ST_PARGEN, l'adresse de la structure ST_ENTDON associée à la donnée à saisir (contenue dans ST_SAISIE), et le numéro de la zone à saisir

Définir chaque élément du plan de saisie associée à chacune des zones à l'aide de e_dfsais(), en précisant les déplacements de zone à zone (haut, bas, droit, gauche) repérés par l'indice de l'élément dans le tableau (un indice négatif déclenche la fin de saisie de la fiche).

B - Saisie proprement dite :

e_saisie() permet de saisir chacune des zones de la fiche, en gérant le déplacement d'une zone à l'autre.

Fonctionnement :

- Si l'on a pas définit de paramètres généraux (parec==NULL), on utilise

une structure ST_PARGEN implicite statique (Cf. ENTREES FORMATEES).

- Si l'on travaille dans des écrans virtuels, il y a mémorisation de l'écran en cours, et ouverture de la fenêtre si l'on change d'écran virtuel (on peut donc saisir des zones situées dans des fenêtres différentes, mais attention aux superpositions !).

- Si l'on a défini une fonction de saisie personnelle (pointeur != NULL), e_saisie() appelle cette fonction; sinon elle appelle e_entdon(). L'une et l'autre actualisent le drapeau de modification associé à la donnée saisie (ste->Modif), et renvoient la touche de sortie de la zone.

- La sortie d'une zone est déclenchée par HAUT, BAS, DROIT en fin de zone ou C_DROIT, GAUCHE en début de zone ou C_GAUCHE.

- La sortie de saisie de la fiche est déclenchée dès que le n° de zone est négatif, ou que la sortie de la dernière zone s'est faite par une autre touche que les précédentes. Le numéro de la dernière zone saisie est mis à jour, ainsi que le drapeau de modification globale de la fiche.

- La valeur de retour est le code de la touche de sortie.

2. La structure ST SAISIE

Une structure ST_SAISIE est propre à une donnée saisie à l'intérieure d'une "fiche".

```
ST_SAISIE
        {

        ST_ENTDON Ste; /* Structure ST_ENTDON associée à la donnée à saisir
                       /* contenu : Cf. ENTREE FORMATTEE

        int (*Saisie)( ST_PARGEN *parec, ST_ENTDON *ste, int numero);
                       /* Pointeur sur la fonction personnelle de saisie
                       /*  ( = NULL en cas d'utilisation de e_entdon() )

        int Haut, Bas, Droit, Gauche;
                       /* Déplacement de zone à zone : indice de la zone
                       /* (struct. ST_SAISIE) dans le tableau (-1 pour SORTIE) */
      };
```

Les "picklists" permettent de choisir un élément dans une liste (triée ou non) affichée dans une fenètre, où chaque élément est repéré par son nom, en déplaçant le curseur à l'aide des flèches et en le sélectionnant avec la touche ENTER.

Les éléments sont quelconques (void *); le nom d'un élément est renvoyé par une fonction "personnelle".

Toutes les fonctions fonctionnent en écran physique et en écran virtuel, mais la "fenêtre" d'une "picklist" est dans l'écran physique.

1. Marche à suivre

1°) Déclarations

- Le pointeur sur la liste des éléments :
  ex : char **list_nom; (pointeur sur vecteur de pointeurs)

- La fonction revoyant le nom d'un élément de la liste :
  ex : char *elt_nom(void *elt);
  Ne pas omettre de définir cette fonction !!!

  N.B. : si la liste est une liste de chaînes (1 élément = 1 chaîne), on peut utiliser la fonction implicite pkl_nom()

- Le pointeur sur la "picklist" : ST_PKL *pkl;

- La palette des couleurs de la "picklist" :
  ex : int palette;

- La fonction associée (facultatif) : elle attend pour argument l'adresse de la picklist, et renvoie un code qui, s'il est **positif ou nul** est traité par la fonction pkl_fct(). Cette fonction peut par exemple donner des précision sur l'élément pointé.
  ex : int fct_ass(ST_PKL *pkl);

2°) Création de la "picklist"

```
/* création de la palette de couleurs */

palette = p_crepal(  31,  /* Couleur du titre et du cadre de la "picklist" */
                    127,  /* Couleur de la fenêtre        ..............  */
                     95,  /* Couleur de l'élément pointé                  */
                     0, 0, 0);  /* Autres couleurs non utilisées          */
```

```
/* création de la "picklist"  (Cf. pkl_cre()) */

pkl = pkl_cre( " PICKLIST ",
               list_nom,
               NULL,      /* Utilisation de la fonction implicite pkl_nom() */
               10, 30,    /* Nb. d'élts et taille du vecteur de pointeurs   */
               5, 30,     /* Position du coin haut-gauche du cadre          */
               10, 1, 20, /* Caractéristiques de la fenêtre :
                             10 lignes, 1 colonne de 20 caractères          */
               palette,   /* Palette des couleurs                           */
               OUI,       /* Ajout possible dans la liste                   */
               OUI,       /* Liste triée par ordre alphabétique             */
               NULL       /* Pas de fonction associée                       */
             );
```

3°) Utilisation de la "picklist"

Le choix d'un élément dans la liste se fait grace à la fonction pkl_fct() :

```
pkl_zon( pkl ); /* Mémoriser la zone écran avant la "picklist" */

choix = pkl_fct( pkl ); /* choix = n° de l'élément sélectionné */

pkl_eff( pkl ); /* Effacer la "picklist" et remettre l'écran
                   à l'état  avant "picklist"                  */
```

2. Fonctions PICKLIST

2.1 Fonctions essentielles

- pkl_cre() : Création d'une "picklist" (allocation dynamique).
- pkl_zon() : Mémorise la zone de l'écran où sera affichée la "picklist"
- pkl_eff() : Efface la "picklist" spécifiée et remet la zone dans son état initial (si toutefois on n'a pas omis de le mémoriser auparavant grâce à pkl_zon() !!).
- pkl_fct() : Fonction PICKLIST

2.2 Autres fonctions

- pkl_aff() : Affichage d'une pickliste

- pkl_tit() : Change le titre d'une pickliste

- pkl_rch() : Recherche (insertion/délétion) d'un élément dans la liste.
  Cet élément doit être du même type que ceux de la liste.

- pkl_pkl() : Construction d'une "picklist" à partir d'une "picklist" source.

- pkl_cat() : Concaténation de deux "picklist"

- pkl_lib() : Destruction d'une "picklist"

3. Structure PICKLIST

```
ST_PKL
     {
     char *Titre;                /* Titre de la "picklist"

     void **Liste;               /* Adresse de la liste des éléments
                                    (tableau de pointeurs)

     char *(*Nom)(void *elt);    /* Fonction "personnelle" renvoyant
                                    le nom d'un élement de la liste

     int Nbelt;                  /* Nb d'éléments dans la liste

     int Nbmax;                  /* Taille actuelle du tableau (de la liste)

     int Elt;                    /* Adresse de l'élément pointé
     int Lig, Col;               /* Coordonnées de l'élt. pointé
                                    dans la fenêtre "picklist" actuelle

     int Lchg, Cchg;             /* Coordonnées coin Haut Gauche du cadre
                                    de la "picklist"
     int Nlig, Ncol, Ncar;       /* Nb de lignes, de colonnes et de caractères
                                    par colonne, de la fenêtre "picklist"

     int Couleur[7];             /* Palette des couleurs de la "picklist"

     char Ajout;                 /* Drapeau d'ajout possible dans la liste
                                                    (OUI/NON)
     char Tri;                   /* Drapeau de tri ALPHA de la liste (OUI/NON)*/

     void (*Fct)(ST_PKL *);      /* Fonction "associée" à pkl_fct()
                                    elle peut par ex. donner des précisions
                                    sur l'élément pointé
     struct P_MEMZON Memzon;     /* Mémoire de la zone de l'écran avant
                                    affichage de la "picklist"

     };
```

Nom: e_affdon - Affichage formaté d'une donnée de type quelconque

Usage: void e_affdon(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Affichage de la donnée spécifiée dans 'ste'

Structure de e_affdon() :

- Convertir la donnée en chaine
  -> ste->Format->Cvd() (si != NULL)
- Recadrer cette chaine
  -> ste->Format->Cad() (si != NULL)
- Afficher la chaine en Vlnorm

Voir aussi: e_zon???(), e_fmt???()

Exemple:

```
ST_ENTDON ste;
float donnée;
int palette;

palette = p_crepal( 0, 112, 127, 124, 0, 0);

/* Initialisation de la structure ST_ENTDON */
e_zonflt( &ste, NULL, palette, 10, 30, &donnee, 6);

/* Affichage proprement dîte */
e_affdon( ste);
```

Nom: e_cad_a - Recadrage d'une chaîne alpha-numérique

Usage: void e_cad_a(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Recadrage final d'une donnée de type chaîne, par élimination des BLANCS en fin et début de chaîne.

- 'ste' : structure ST_ENTDON associée à la chaîne saisie

Le recadrage s'effectue sur la chaîne de travail (ste->Chaine)

Nom: e_cad_e - Recadrage d'un entier (type INT ou UNSIGNED ou LONG)

Usage: void e_cad_e(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Recadrage d'une donnée de type INT ou UNSIGNED ou LONG
- 'ste' : adresse de la structure ST_ENTDON associée

Recadrage à droite, en respectant le "format" de l'entier :
- nombre de chiffres (= longueur de la zone)

Nom: e_cad_r - Recadrage d'un réel (type FLOAT ou DOUBLE)

Usage: void e_cad_r(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Recadrage d'une donnée de type FLOAT ou DOUBLE
- 'ste' : adresse de la structure ST_ENTDON associée

Recadrage à droite, en respectant le "format" du réel :
- nombre de décimales et nombre de chiffres avant la virgule si 'ste->Format->Ptr' != NULL
- nombre de chiffres significatif (= longueur de la zone) sinon

Nom: e_cvd_c - Fonction de Conversion Début pour une donnée de type CARACTERE

Usage: char *e_cvd_c(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: Adresse d'une chaîne contenant le caractère à saisir en première position

Description: Alloue une chaîne de 2 octets (par calloc()) et place le caractère à saisir en tête (ste->Donnée)

- 'ste' : structure ST_ENTDON associée au caractère à saisir

Nom: e_cvd_d - Fonction de Conversion Début pour une donnée de type DOUBLE

Usage: char *e_cvd_d(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: adresse de la chaîne de travail

Description: La fonction alloue une chaîne de longueur 'ste->Lg'+1 (par calloc()) et convertit la donnée ('ste->Donnee') en chaîne

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvd_dt - Fonction de Conversion Début pour une donnée de type DATE

Usage: char *e_cvd_dt(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: adresse de la chaîne de travail

Description: La fonction alloue une chaîne (calloc()) de longueur 'ste->Lg'+1 (longueur du format de la date).
Si la donnée est vide (ste->Donnee[0]==VIDE), on recopie le format dans chaîne et dans la donnée, puis on remplace dans la donnée tous les caractères 'J', 'M' et 'A' par '0'.
Sinon, on recopie la donnée dans la chaîne.

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvd_f - Fonction de Conversion Début pour une donnée de type FLOAT

Usage: char *e_cvd_f(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: adresse de la chaîne de travail

Description: La fonction alloue une chaîne de longueur 'ste->Lg'+1 (par calloc()) et convertit la donnée ('ste->Donnee') en chaîne

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvd_i - Fonction de Conversion Début pour une donnée de type INT

Usage: char *e_cvd_i(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: adresse de la chaîne de travail

Description: La fonction alloue une chaîne de longueur 'ste->Lg'+1 (par calloc()) et convertit la donnée ('ste->Donnee') en chaîne

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvd_l - Fonction de Conversion Début pour une donnée de type LONG

Usage: char *e_cvd_l(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: adresse de la chaîne de travail

Description: La fonction alloue une chaîne de longueur 'ste->Lg'+1 (par calloc()) et convertit la donnée ('ste->Donnee') en chaîne

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvd_u - Fonction de Conversion Début pour une donnée de type UNSIGNED

Usage: char *e_cvd_u(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: adresse de la chaîne de travail

Description: La fonction alloue une chaîne de longueur 'ste->Lg'+1 (par calloc()) et convertit la donnée ('ste->Donnee') en chaîne

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvf_c - Fonction de Conversion Fin pour une donnée de type CARACTERE

Usage: void e_cvf_c(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: copie le premier caractère de la chaîne de travail (ste->Chaine) dans le caractère à saisir

Nom: e_cvf_d - Fonction de Conversion Fin pour une donnée de type DOUBLE

Usage: void e_cvf_d(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: La fonction convertit la chaîne de travail (ste->Chaine) en DOUBLE, qui est recopié dans la donnée

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvf_dt - Fonction de Conversion Fin pour une donnée de type DATE

Usage: void e_cvf_dt(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: La fonction convertit la chaîne de travail (ste->Chaine) en DATE, en respectant le format, et la recopie dans la donnée

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvf_f - Fonction de Conversion Fin pour une donnée de type FLOAT

Usage: void e_cvf_f(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: La fonction convertit la chaîne de travail (ste->Chaine) en FLOAT, qui est recopié dans la donnée

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvf_i - Fonction de Conversion Fin pour une donnée de type INT

Usage: void e_cvf_i(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: La fonction convertit la chaîne de travail (ste->Chaine) en INT, qui est recopié dans la donnée

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvf_l - Fonction de Conversion Fin pour une donnée de type LONG

Usage: void e_cvf_l(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: La fonction convertit la chaîne de travail (ste->Chaine) en LONG, qui est recopié dans la donnée

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_cvf_u - Fonction de Conversion Fin pour une donnée de type UNSIGNED

Usage: void e_cvf_u(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: aucune

Description: La fonction convertit la chaîne de travail (ste->Chaine) en UNSIGNED, qui est recopié dans la donnée

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Nom: e_entalp - saisie d'une chaîne de caractères

Usage:
```
int  e_entalp( ST_PARGEN *parec,
               int ecran, int lig, int col,
               char *donnee, int lg, int *modif)
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de la zone de saisie

Description: saisie d'une chaîne de caractères, selon les modalités décrites dans 'parec' (si != NULL)

N.B. : les structures ST_FORMAT et ST_ENTDON nécessaires à e_entdon() sont générées automatiquement

- 'parec' : adresse de la structure ST_PARGEN contenant les paramètres généraux du mode de saisie. Si parec == NULL : utilisation de paramètres standards (pas de sortie gauche et droite, pas de départ automatique en fin de zone, pas de touche d'entrée dans la zone, pas de touche d'initialisation ni de touche d'effacement (Cf. e_pargen())
- 'ecran' : n° d'écran virtuel, ou de palette dans l'écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir ne pas oublier d'initialiser la donnée avant la saisie
- 'lg' : longueur de la chaîne (= longueur de la zone de saisie)
- 'modif' : adresse du drapeau de modification au retour, modif = OUI si la donnée a été modifiée

Voir aussi: e_entdon(), e_zonalp(), e_fmtdon(), e_pargen()

Exemple:
```
char chaîne[21];
int palette;
int modif;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
memset( chaine, VIDE, 21);
e_entalp( NULL, palette, 10, 40, chaine, 20, &modif);
```

Nom: e_entcar - saisie d'un caractère

Usage:
```
int  e_entcar( ST_PARGEN *parec,
               int ecran, int lig, int col,
               char *donnee, char *liste, int *modif)
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de la zone de saisie

Description: saisie d'un caractère, selon les modalités décrites dans 'parec' (si != NULL)

N.B. : les structures ST_FORMAT et ST_ENTDON nécessaires à e_entdon() sont générées automatiquement

- 'parec' : adresse de la structure ST_PARGEN contenant les paramètres généraux du mode de saisie. Si parec == NULL : utilisation de paramètres standards (pas de sortie gauche et droite, pas de départ automatique en fin de zone, pas de touche d'entrée dans la zone, pas de touche d'initialisation ni de touche d'effacement (Cf. e_pargen())
- 'ecran' : n° d'écran virtuel, ou de palette dans l'écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir ne pas oublier d'initialiser la donnée avant la saisie
- 'liste' : liste des caractères acceptés cette liste est mise dans le pointeur "boite aux lettres" et est utilisée par le 'filtre des caractères valides' (Cf. e_fcv_c()).
- 'modif' : adresse du drapeau de modification au retour, modif = OUI si la donnée a été modifiée

Voir aussi: e_entdon(), e_zoncar(), e_fmtcar(), e_pargen()

Exemple:
```
char car;
int palette;
int modif;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
car = VIDE;
e_entcar( NULL, palette, 10, 40, &car, "HhFf", &modif);
```

Nom: e_entdat - saisie d'une donnée de type DATE

Usage:
```
int e_entdat( ST_PARGEN *parec,
              int ecran, int lig, int col,
              char *donnee, char *format,
              int *modif)
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de la zone de saisie

Description: saisie d'une donnée de type DATE (chaîne de caractères), selon les modalités décrites dans 'parec' (si != NULL)
N.B. : les structures ST_FORMAT et ST_ENTDON nécessaires à e_entdon() sont générées automatiquement

- 'parec' : adresse de la structure ST_PARGEN contenant les paramètres généraux du mode de saisie. Si parec == NULL : utilisation de paramètres standards (pas de sortie gauche et droite, pas de départ automatique en fin de zone, pas de touche d'entrée dans la zone, pas de touche d'initialisation ni de touche d'effacement (Cf. e_pargen())
- 'ecran' : n° d'écran virtuel, ou de palette dans l'écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la date à saisir ne pas oublier d'initialiser la donnée avant la saisie
- 'format' : format de la date, du type "JJ/MM/AA" ou "jour : JJ mois : MM" ou "AAAAMMJJ"
  . la longueur du format donne la taille de la zone
  . tous les caractères autres que 'J', 'M' et 'A' sont traités comme du texte
- 'modif' : adresse du drapeau de modification au retour, modif = OUI si la donnée a été modifiée

Remarque : si 'donnee' est vide, il y a recopie du format dans la chaîne initiale

Voir aussi: e_entdon(), e_zondat(), e_fmtdat(), e_pargen()

Exemple:
```
char date[21];
int palette, modif;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
memset( date, VIDE, 21);
e_entdat( NULL, palette, 10, 40, date, "Né le : JJ-MM-AAAA", &modif);
```

Nom: e_entdbl - saisie d'une donnée de type DOUBLE

Usage:
```
int e_entdbl( ST_PARGEN *parec,
              int ecran, int lig, int col,
              double *donnee, char *format,
              int *modif)
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de la zone de saisie

Description: saisie d'une donnée de type DOUBLE, selon les modalités décrites dans 'parec' (si != NULL)
N.B. : les structures ST_FORMAT et ST_ENTDON nécessaires à e_entdon() sont générées automatiquement

- 'parec' : adresse de la structure ST_PARGEN contenant les paramètres généraux du mode de saisie.
  Si parec == NULL : utilisation de paramètres standards (pas de sortie gauche et droite, pas de départ automatique en fin de zone, pas de touche d'entrée dans la zone, pas de touche d'initialisation ni de touche d'effacement (Cf. e_pargen())
- 'ecran' : n° d'écran virtuel, ou de palette dans l'écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir
  ne pas oublier d'initialiser la donnée avant la saisie
- 'format' : format de la donnée, du type "##.#" ou "+##.#" ou "-##.#"

  il permet de décrire la taille de la zone, le nombre de décimales et de chiffres avant la virgule, ainsi que les bornes mini et maxi
  le '+' et '-' imposent le signe du réel
  ex : "##.#" -> longueur = 4
  -> 1 décimale, 2 chiffres avant la virgule
  -> maxi = 99.9
  -> mini = -9.9
- 'modif' : adresse du drapeau de modification
  au retour, modif = OUI si la donnée a été modifiée

Voir aussi: e_entdon(), e_zondbl(), e_fmtdbl(), e_pargen()

Exemple:
```
double donnee = 0;
int palette, modif;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
e_entdbl( NULL, palette, 10, 40, &donnee, "+##.#", &modif);
```

Nom: e_entdon - Entrée formattée pour une donnée de type quelconque

Usage: int e_entdon(ST_PARGEN *parec, ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de saisie de la zone

Description: Saisie la donnée décrite dans 'ste'
selon les modalités fournie par 'parec' et 'ste' .
Au retour ste->Modif = OUI si la donnée a été modifiée.
Retourne la touche de sortie.

- 'parec' : structure PARGEN utilisée (si NULL, e_entdon() utilisera des paramètres standards)

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Structure de e_entdon() :

- Remplacement des pointeurs NULL par des paramètres standards ('parec', 'ste->Format', ...)

- Conversion de la donnée en chaine de caractères
  -> ste->Format->Cvd() (si != NULL)

- Affichage de la chaîne, en Vlmodif si parec->T_mod == 0, en Vlnorm sinon

- TANT QUE la donnée n'est pas correcte :

  - Appel de e_fentre()
  - Conversion de la chaine en donnée
    -> ste->Format->Cvf() (si != NULL)
  - Vérification de la donnée
    -> ste->Format->Ver() (si != NULL)

- Recadrage de la chaine
  -> ste->Format->Cad() (si != NULL)

- Affichage de la chaine en Vlnorm, si on est entré dans la zone de saisie

- Libération de la chaîne de travail allouée par la fonction de conversion début
  (si ste->Chaine != ste->Donnee)

- RETOURNE la touche de sortie renvoyée par e_fentre()

Remarques :

- *Si parec == NULL, création d'une struct. PAR_GENE standard (tous les paramètres à 0 ou NON, sauf car. remp. = BLANC)*
- *Si ste->Format == NULL, création d'une struct. ST_FORMAT standard, ne comportant que des pointeurs NULL.*
- *Si ste->Format->Fcv() == NULL, ste->Format->Fcv = e_fcv_st (filtre des caractères valides standard)*
- *Si ste->Format->Fts() == NULL, ste->Format->Fts = e_fts_st (filtre des touches spécifiques standard)*

Voir aussi: e_fmt???(), e_zon???(), e_fentre()

Exemple: Cf. listing du programme testlist.c

```
ST_PARGEN parec;
ST_ENTDON ste;
ST_FORMAT fmt;
float donnée = 0.;
int palette, sortie;

palette = p_crepal( 0, 112, 127, 124, 0, 0);

/* Initialisation des paramètres généraux */
e_pargen( &parec, OUI, OUI, OUI, BLANC, 0, ESC, 0);

/* Initialisation du format */
e_fmtflt( &fmt, 5, 0, 10000., 15000., 0., 30000.,
                "Trop faible", "Trop élevé");

/* Initialisation de la structure ST_ENTDON */
e_zonflt( &ste, &fmt, palette, 10, 30, &donnee, 6);

/* Saisie proprement dîte */
sortie = e_entdon( &parec, &ste);
```

Nom: e_entfic - Entrée d'un nom de fichier ou de répertoire

Usage:
```
int  e_entfic( ST_PARGEN *parec,
               int ecran, int lig, int col,
               char *donnee, int lg, int *modif
               int *nature )
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de la zone de saisie

Description:

1°) Saisie du nom du fichier (Cf. e_entalp())
2°) Vérification (Cf. dir_ver())

- 'parec', 'ecran', 'lig', 'col': Cf. e_entalp()

- 'donnee' : adresse de la chaîne contenant le nom du fichier à saisir.
  la chaîne doit être dimensionnée à MAXPATH (Cf. dir.h), car au retour, donnee contient le nom complet
  lecteur + path + nom + ext
- 'lg' : longueur de la zone de saisie (lg <= MAXPATH en général)
- 'nature' : au retour, ce drapeau indique le type de fichier défini dans GRDIR.H :
  FIC_INC : fichier inconnu
  FICHIER : fichier existant
  REPERT : répertoire
  JOKER : présence de '*' ou '?'

N.B. : e_entfic() vérifie si le répertoire est correct : au cas où il n'existe pas, message signalé par ernfatal().

Voir aussi: e_entalp(), e_pargen(), dir_ver(), dir_???()

Exemple:
```
char fic_nom[MAXPATH];
int palette;
int fic_nat;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
memset( chaine, VIDE, MAXPATH);
e_entfic( NULL, palette, 10, 40, fic_nom, 20, &fic_nat);
```

Nom: e_entflt - saisie d'une donnée de type FLOAT

Usage:
```
int e_entflt( ST_PARGEN *parec,
              int ecran, int lig, int col,
              float *donnee, char *format,
              int *modif)
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de la zone de saisie

Description: saisie d'une donnée de type FLOAT, selon les modalités décrites dans 'parec' (si != NULL)
N.B. : les structures ST_FORMAT et ST_ENTDON nécessaires à e_entdon() sont générées automatiquement

- 'parec' : adresse de la structure ST_PARGEN contenant les paramètres généraux du mode de saisie.
  Si parec == NULL : utilisation de paramètres standards (pas de sortie gauche et droite, pas de départ automatique en fin de zone, pas de touche d'entrée dans la zone, pas de touche d'initialisation ni de touche d'effacement (Cf. e_pargen())
- 'ecran' : n° d'écran virtuel, ou de palette dans l'écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir
  ne pas oublier d'initialiser la donnée avant la saisie
- 'format' : format de la donnée, du type "##.#" ou "+##.#" ou "-##.#"

  il permet de décrire la taille de la zone, le nombre de décimales et de chiffres avant la virgule, ainsi que les bornes mini et maxi
  le '+' et '-' imposent le signe du réel
  ex : "##.#" -> longueur = 4
  -> 1 décimale, 2 chiffres avant la virgule
  -> maxi = 99.9
  -> mini = -9.9
- 'modif' : adresse du drapeau de modification
  au retour, modif = OUI si la donnée a été modifiée

Voir aussi: e_entdon(), e_zonflt(), e_fmtflt(), e_pargen()

Exemple:
```
float donnee = 0;
int palette, modif;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
e_entflt( NULL, palette, 10, 40, &donnee, "+##.#", &modif);
```

Nom: e_entint - saisie d'une donnée de type INT

Usage:
```
int e_entint( ST_PARGEN *parec,
              int ecran, int lig, int col,
              int *donnee, char *format,
              int *modif)
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de la zone de saisie

Description: saisie d'une donnée de type INT, selon les modalités décrites dans 'parec' (si != NULL)
N.B. : les structures ST_FORMAT et ST_ENTDON nécessaires à e_entdon() sont générées automatiquement

- 'parec' : adresse de la structure ST_PARGEN contenant les paramètres généraux du mode de saisie.
  Si parec == NULL : utilisation de paramètres standards (pas de sortie gauche et droite, pas de départ automatique en fin de zone, pas de touche d'entrée dans la zone, pas de touche d'initialisation ni de touche d'effacement (Cf. e_pargen())
- 'ecran' : n° d'écran virtuel, ou de palette dans l'écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir
  ne pas oublier d'initialiser la donnée avant la saisie
- 'format' : format de la donnée, du type "###" ou "+###" ou "-###"
  le nombre de '#' donne la taille de la zone, ainsi que les bornes mini et maxi
  le '+' et '-' imposent le signe de l'entier
  ex : "##" -> longueur = 2
  -> maxi = 99
  -> mini = -9
- 'modif' : adresse du drapeau de modification
  au retour, modif = OUI si la donnée a été modifiée

Voir aussi: e_entdon(), e_zonint(), e_fmtint(), e_pargen()

Exemple:
```
int donnee = 0;
int palette, modif;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
e_entint( NULL, palette, 10, 40, &donnee, "+##", &modif);
```

Nom: e_entlng - saisie d'une donnée de type LONG

Usage:
```
int e_entlng( ST_PARGEN *parec,
              int ecran, int lig, int col,
              long *donnee, char *format,
              int *modif)
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de la zone de saisie

Description: saisie d'une donnée de type LONG, selon les modalités décrites dans 'parec' (si != NULL)
N.B. : les structures ST_FORMAT et ST_ENTDON nécessaires à e_entdon() sont générées automatiquement

- 'parec' : adresse de la structure ST_PARGEN contenant les paramètres généraux du mode de saisie. Si parec == NULL : utilisation de paramètres standards (pas de sortie gauche et droite, pas de départ automatique en fin de zone, pas de touche d'entrée dans la zone, pas de touche d'initialisation ni de touche d'effacement (Cf. e_pargen())
- 'ecran' : n° d'écran virtuel, ou de palette dans l'écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir ne pas oublier d'initialiser la donnée avant la saisie
- 'format' : format de la donnée, du type "###" ou "+###" ou "-###" le nombre de '#' donne la taille de la zone, ainsi que les bornes mini et maxi
  ex : "##" -> longueur = 2
  -> maxi = 99
  -> mini = -9
- 'modif' : adresse du drapeau de modification au retour, modif = OUI si la donnée a été modifiée

Voir aussi: e_entdon(), e_zonlng(), e_fmtlng(), e_pargen()

Exemple:
```
long donnee = 0;
int palette, modif;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
e_entlng( NULL, palette, 10, 40, &donnee, "##", &modif);
```

Nom: e_enttxt - Entrée formatée pour une donnée de type TEXTE
(tableau de chaines de caractères)

Usage:
```
int  e_enttxt( ST_PARGEN *parec,
               int ecran, int lig, int col,
               char texte[][], int nlig, int ncar,
               int *modif, ST_FORMAT *fmt);
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de saisie du texte

Description: Saisie du texte 'texte' selon les modalités fournies par 'parec' et 'fmt'.
Au retour 'modif' = OUI si le texte a été modifié.
Retourne la touche de sortie.

- 'parec' : structure ST_PARGEN utilisée (si NULL, e_enttxt() utilisera des paramètres standards).
- 'ecran' : n° d'écran VSI ou n° de palette
- 'lig' :
- 'col' : coordonnées du début de la zone de texte
- 'texte' : adresse du texte (tableau de chaines prédimensionné : char texte[<nlig>][<ncar+1>])
  Ne pas oublier de l'initialiser avant saisie !
- 'nlig' : nombre de lignes de texte
- 'ncar' : nombre de caractères/ligne (sans le VIDE)
- 'modif' : drapeau de modification
- 'fmt' : adresse d'une structure ST_FORMAT décrivant les modalités de saisie du texte (ou NULL). e_enttxt() utilise sa propre structure ST_FORMAT, qui "héritera" des pointeurs non NULL de 'fmt', ce qui permet de personnaliser la saisie de texte

Fonctionnement de e enttxt() :

e_enttxt() fonctionne en écran virtuel ou en écran physique (compilation conditionnelle). En écran physique, attention au débordement !

e_enttxt() **n'appelle pas e_entdon()** et utilise ses propres structures ST_PARGEN, ST_ENTDON, et ST_FORMAT.

1) Initialisation des structures :

   - initialisation d'une structure ST_ENTDON associée

à la zone de texte

- initialisation d'une structure ST_PARGEN propre à la saisie de chacune des lignes du texte :
  . sorties DROITE, GAUCHE, et en fin de ligne
  . caractère de remplissage = BLANC
  . touches spéciales = celles de 'parec' (ou 0)

- initialisation d'une structure ST_FORMAT décrivant les modalités de saisie du texte :
  . si 'fmt' != NULL, "héritage" des pointeurs non NULL de 'fmt', ou utilisation de pointeurs implicites
  . si 'fmt' == NULL, utilisation de pointeurs implicites
    Pas de Conversion début et fin
    Pas de Vérification
    Pas de Recadrage
    Filtres des caractères valides e_fcv_st()
    Filtres des touches spécifiques e_fcv_st()
    Pas de Pointeur "boîte aux lettres"

2) Affichage du texte : en Vlnorm ou Vlmodif
   suivant 'parec->T_mod'
   En écran virtuel, mémorisation de la position initiale de la fenêtre

3) Boucle de saisie ligne/ligne :

   TANT QU'on ne sort pas de la saisie

   - Recopie de la ligne de texte en cours dans la chaine de travail

   - Appel de e_fentre()

   - Si la ligne a été modifiée :

     . Vérification de la ligne
       -> ste->Format->Ver() (si != NULL)
     . Actualisation du drapeau global 'modif'
     . Recopie de la chaîne de travail dans la ligne

   - Traitement des touches "éditeur de texte" :
     ces touches correspondent à une "sortie" pour la saisie d'une ligne

     . ENTER, DROIT : passage au début de la ligne suivante, ou sortie (fin de boucle) si on est sur la dernière ligne et que 'parec->Sortied' = OUI

     . BACKSPACE, GAUCHE : passage à la fin de la ligne précédente, ou sortie (fin de boucle) si on est sur la première ligne et que 'parec->Sortieg' = OUI

     . BAS : passage à la ligne suivante, au même niveau ou en fin de ligne, ou sortie (fin de boucle) si on est sur la dernière ligne et que

'parec->Sortied' = OUI

. HAUT : passage à la ligne précédente, au même niveau ou en fin de ligne, ou sortie (fin de boucle) si on est sur la première ligne et que 'parec->Sortieg' = OUI

. PRECEDE : retour 10 lignes en arrière, au même niveau ou en fin de ligne, ou sortie (fin de boucle) si on est sur la première ligne et que 'parec->Sortieg' = OUI

. SUITE : passage 10 lignes en avant, au même niveau ou en fin de ligne, ou sortie (fin de boucle) si on est sur la première ligne et que 'parec->Sortied' = OUI

. CTR_Y : suppression de la ligne en cours, affichage du texte modifié

. CTR_N : insertion d'une ligne et décalage à partir de la ligne en cours, affichage du texte modifié

. Autre touche : sortie (fin de boucle)

4) Elimination des blancs en fin de ligne et affichage du texte en Vlnorm
en écran virtuel, on replace la fenêtre dans sa position initiale

5) Libération de la chaîne de travail allouée

RETOURNE la touche de sortie renvoyée par e_fentre()

Remarques :
. Si parec == NULL, aucune "sortie" n'est autorisée
. L'utilisation d'une structure ST_FORMAT permet notamment la Vérification de chacune des lignes saisies et l'utilisation de "picklist"

Voir aussi: e_fcv_st(), e_fts_tx(), e_fentre()

Exemple:

```
char texte[10][61]; /* Texte de 10 lignes x 60 caractères */
ST_PARGEN parec;
int palette, modif, sortie;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
memset( texte, VIDE, 10*61);

/* Initialisation des paramètres généraux */
e_pargen( &parec, OUI, OUI, OUI, BLANC, 0, ESC, 0);

/* Saisie proprement dîte */
```

```
sortie = e_entdon( &parec, palette, 5, 10,
                   texte, 10, 60, &modif, NULL);
```

Nom: e_entuns - saisie d'une donnée de type UNSIGNED

Usage:
```
int e_entuns( ST_PARGEN *parec,
              int ecran, int lig, int col,
              unsigned *donnee, char *format,
              int *modif)
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie de la zone de saisie

Description: saisie d'une donnée de type UNSIGNED, selon les modalités décrites dans 'parec' (si != NULL)
N.B. : les structures ST_FORMAT et ST_ENTDON nécessaires à e_entdon() sont générées automatiquement

- 'parec' : adresse de la structure ST_PARGEN contenant les paramètres généraux du mode de saisie.
  Si parec == NULL : utilisation de paramètres standards (pas de sortie gauche et droite, pas de départ automatique en fin de zone, pas de touche d'entrée dans la zone, pas de touche d'initialisation ni de touche d'effacement (Cf. e_pargen())
- 'ecran' : n° d'écran virtuel, ou de palette dans l'écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir
  ne pas oublier d'initialiser la donnée avant la saisie
- 'format' : format de la donnée, du type "###"
  le nombre de '#' donne la taille de la zone, ainsi que les bornes mini et maxi
  ex : "##" -> longueur = 2
  -> maxi = 99
  -> mini = 0
- 'modif' : adresse du drapeau de modification
  au retour, modif = OUI si la donnée a été modifiée

Voir aussi: e_entdon(), e_zonuns(), e_fmtuns(), e_pargen()

Exemple:
```
unsigned donnee = 0;
int palette, modif;

palette = p_crepal( 0, 112, 127, 124, 0, 0);
e_entuns( NULL, palette, 10, 40, &donnee, "##", &modif);
```

Nom: e_fcv_c - Filtre des caractères valides pour la saisie d'une donnée de type CARACTERE

Usage: int e_fcv_c(int *s, ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- CAR_OK : caractère accepté
- INCORRECT : caractère refusé
- A_TRAITER : à passer dans les filtres suivants (Cf. e_fentre())

Description: La fonction renvoie A_TRAITER pour tous les caractères non-imprimables

(code '*s' < 32 ou > 255)

Pour les caractères imprimables (31 < '*s' <256)

- s'il existe une liste de caractères acceptables (liste = (char *) ste->Ptr), la fonction teste la présence de '*s' dans cette liste, et renvoie CAR_OK si le test est positif, INCORRECT sinon

- s'il n'existe pas de liste (ste->Ptr==NULL), la fonction renvoie CAR_OK

Voir aussi: e_fcv_st()

Nom: e_fcv_dt - Filtre des caractères valides pour la saisie d'une donnée de type DATE

Usage: int e_fcv_dt(int *s, ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- CAR_OK : caractère accepté
- A_TRAITER : à passer dans les filtres suivants (Cf. e_fentre())

Description: La fonction n'accepte la saisie de caractères (chiffre ou ' ') que dans les "sous-zones" jour ("JJ"), mois ("MM") et année ("AA" ou "AAAA") et passe à la "sous-zone" suivante si le curseur est placé à la fin d'une "sous-zone".
Sinon, elle retourne 'A_TRAITER'.

Voir aussi: e_fts_dt()

Nom: e_fcv_e - Filtre des caractères valides pour la saisie d'un entier de type INT ou UNSIGNED ou LONG

Usage: int e_fcv_e(int *s, ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- CAR_OK : caractère accepté
- INCORRECT : caractère refusé
- A_TRAITER : à passer dans les filtres suivants (Cf. e_fentre())

Description: La fonction accepte :
- tous les chiffres et le blanc ('0' <= '*s' <= '9' ou *s== ' ')
- le signe '+' ou '-' en premier caractère

Les autres caractères imprimables sont INCORRECT
Les autres doivent passer dans les filtres suivants (A_TRAITER)

Voir aussi: e_fcv_r()

Nom: e_fcv_r - Filtre des caractères valides pour la saisie d'un réel de type FLOAT ou DOUBLE

Usage: int e_fcv_r(int *s, ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- CAR_OK : caractère accepté
- INCORRECT : caractère refusé
- A_TRAITER : à passer dans les filtres suivants (Cf. e_fentre())

Description: La fonction accepte :
- tous les chiffres et le blanc ('0' <= '*s' <= '9' ou *s== ' ')
- le signe '+' ou '-' en premier caractère
- le point décimal (ou la virgule) que si c'est le seul

Les autres caractères imprimables sont INCORRECT
Les autres doivent passer dans les filtres suivants (A_TRAITER)

Voir aussi: e_fcv_e()

Nom: e_fcv_st - Filtre des caractères valides standard
(valeur implicite si fmt->Fcv == NULL)

Usage: int e_fcv_st(int *s, ST_ENDON *ste);

Prototype dans: grentre.h

Valeur retournée: CAR_OK : caractère accepté
A_TRAITER : caractère à passer dans les filtres suivants

Description: Accepte tous les caractères imprimables (31 < '*s' < 256)
Sinon retourne A_TRAITER

- 's' : code du caractère entré au clavier
(l'adresse permet de le modifier)
- 'ste' : structure ST_ENTDON associée à la donnée
en cours de saisie

Voir aussi: e_fentre(), e_fcv_?()

Nom: e_fts_dt - Filtre des touches spécifiques pour la saisie d'une donnée de type DATE

Usage: int e_fts_dt(int *s, ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- TRAITE : touche traitée par la fonction
- INCORRECT : touche incorrecte
- A_TRAITER : à passer dans les filtres suivants (Cf. e_fentre())
- DE_SORTIE : touche de sortie de la zone de saisie

Description: La fonction gère les déplacements par les flèches DROITE et GAUCHE, DEL et BACKSPACE :
- si on est pas "dans la zone saisie" -> DE_SORTIE
- si on est "dans la zone de saisie" :
  . si on est dans une "sous-zone' ("JJ", "MM", "AA" ou "AAAA"), passage au caractère suivant , ou à la sous-zone suivante -> A_TRAITER
  . sinon, passage à la 1ère "sous-zone" suivante -> A_TRAITER

INS est INCORRECT.
Autres touches, Cf. e_fts_st()

Voir aussi: e_fts_st()

Nom: e_fts_st - Filtre des touches spécifiques standard
(valeur implicite si fmt->Fts == NULL)

Usage: int e_fts_st(int *s, ST_ENDON *ste);

Prototype dans: grentre.h

Valeur retournée:
DE_SORTIE : sortie de la zone de saisie
TRAITE : entrée d'un autre caractère
INCORRECT : entrée d'un autre caractère
A_TRAITER : caractère à passer par le filtre suivant

Description: Traitement standard des "touches spécifiques"

- 's' : code du caractère entré au clavier (l'adresse permet de le modifier)
- 'ste' : structure ST_ENTDON associée à la donnée en cours de saisie

- Touches de sortie :
  ESC, ENTER, CTR_C,
  F1 à F10,
  HAUT, BAS, SUITE, PRECEDE,
  et si on n'est pas dans la zone :
  HOME, END, DROIT, GAUCHE, BACKSPACE, DEL, INS
- Touches à traiter :
  Si on est dans la zone :
  HOME, END, DROIT, GAUCHE, BACKSPACE, DEL, INS
- Touches incorrectes : toutes les autres
- Touches traitées : aucunes (évidemment)

Voir aussi: e_fentre(), e_fts_tx(), e_fts_dt()

Nom: e_fcv_st - Filtre des touches spécifiques pour la saisie d'un texte (Cf. e_enttxt())

Usage: int e_fcv_tx(int *s, ST_ENDON *ste);

Prototype dans: grentre.h

Valeur retournée:
DE_SORTIE : sortie de la zone de saisie
TRAITE : entrée d'un autre caractère
INCORRECT : entrée d'un autre caractère
A_TRAITER : caractère à passer par le filtre suivant

Description: Traitement des "touches spécifiques" pour un texte

- 's' : code du caractère entré au clavier (l'adresse permet de le modifier)
- 'ste' : structure ST_ENTDON associée à la donnée en cours de saisie

- Touches de sortie :
  CTR_N
  CTR_Y (ste->Modif forcé à NON pour éviter Vérification)
  ESC, ENTER, CTR_C,
  F1 à F10,
  CTR_<Touche de Déplacement>
  et si on n'est pas dans la zone :
  HOME, END, DROIT, GAUCHE, BACKSPACE
- Touches à traiter :
  Si on est dans la zone :
  HOME, END, DROIT, GAUCHE, BACKSPACE, DEL, INS
- Touches incorrectes : dans tous les autres cas
- Touches traitées : aucunes

Voir aussi: e_fentre()

Nom: e_fentre - Entrée d'une donnée sous forme de chaîne de caractères

Usage:
```
int e_fentre( ST_PARGEN *parec,
              ST_ENTDON *ste, char *ch_init);
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie

Description: e_fentre() saisie une donnée sous forme de chaine de caractères selon les indications données par les structures 'parec' et 'ste'

- 'parec' : structure ST_PARGEN utilisée (paramètres généraux de saisie)
- 'ste' : structure ST_ENTDON associée à la donnée à saisir
- 'ch_init' : chaîne initiale (pour la ré-initialisation éventuelle de la zone)

Structure de e_fentre() :

TANT QUE on ne sort pas de la zone de saisie (Cf. e_fts_?())

- Attente d'un caractère au clavier.

1) Filtre des caractères valides
   -> ste->Format->Fcv()
   au retour :
   si CAR_OK : caractère accepté provoque l'entrée dans la zone en modif si on n'y était pas (affichage en Vlmodif)
   si INCORRECT : renvoie à la saisie d'un caratère (fin de boucle)
   si A_TRAITER : le caractère passe dans les filtres suivants

2) Traitement des touches "paramètres généraux" :

   - parec->T_mod : entrée dans la zone de saisie pour modification

   si on est dans la zone de saisie

   - parec->T_init: réinitialisation de la zone à sa valeur initiale 'ch_init'

   - parec->T_eff : effacement de la zone

3) Filtre des touches spécifiques :
   -> ste->Format->Fts()
      au retour :
      - si TRAITE
        ou INCORRECT : renvoie à la saisie
                       d'un caractère (fin de boucle)
      - si DE_SORTIE : valide la saisie et
                       sort de e_fentre()
      - si A_TRAITER : le caractère passe
                       dans le filtre suivant

4) Traitement des touches d'édition : ne fonctionnent
   que si l'on est **dans la zone de saisie**

  . HOME : curseur en début de zone
  . END   : curseur en fin de zone
  . DROIT : déplacement d'un caractère vers la droite
            en fin de zone, si parec->Sortied == OUI,
            -> sortie de zone, donc de e_fentre()
  . GAUCHE: déplacement d'un caractère vers la gauche
            en début de zone, si parec->Sortieg == OUI,
            -> sortie de zone, donc de e_fentre()
  . BACKSPACE: effacement du caractère avant le curseur
            en début de zone, si parec->Sortieg == OUI,
            -> sortie de zone, donc de e_fentre()
  . DEL   : effacement du caractère sous le curseur
  . INS   : bascule insertion / recouvrement
            modifie la taille du curseur

RETOURNE la touche de sortie

Voir aussi: e_entdon(), e_fcv_?(), e_fts_?()

Nom: e_fmtalp - initialisation d'une structure ST_FORMAT pour la saisie d'une chaîne de caractères

Usage: void e_fmtalp(ST_FORMAT *fmt)

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_FORMAT associée à la chaîne à saisir repérée par son adresse 'fmt' :

- 'fmt' : adresse de la structure ST_FORMAT.
  - pas de fonctions de conversion début et fin
  - recadrage à guche par la fonction e_cad_a()
  - filtre des caractères valides standard e_fcv_st
  - filtre des touches spécifiques standard e_fts_st

Voir aussi: e_entalp(), e_fmtdon(), e_cad_a(), e_fcv_st(), e_fts_st()

Nom: e_fmtcar - initialisation d'une structure ST_FORMAT pour la saisie d'une chaîne de caractères

Usage:
```
void e_fmtcar( ST_FORMAT *fmt,
               char *liste);
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_FORMAT repérée par son adresse 'fmt' :

- 'fmt' : adresse de la structure ST_FORMAT associée au caractère saisi. Cette structure est initialisée avec les pointeurs suivants :
  - e_cvd_c et e_cvf_c pour les conversions début et fin
  - e_fcv_c pour le filtre des caractères valides
  - e_fts_st pour le filtre des touches spécifiques
  - 'liste' pour le pointeur "boîte aux lettres"
- 'liste' : liste des caractères acceptés (chaine constante ou statique)

Voir aussi: e_entcar(), e_zoncar(), e_cvd_c(), e_cvf_c(), e_fcv_c(), e_fts_st()

Exemple:
```
ST_FORMAT fmt;

e_fmtcar( &fmt, "FfMm");
```

Nom: e_fmtdat - initialisation d'une structure ST_FORMAT pour une donnée de type DATE

Usage:
```
void e_fmtdat( ST_ENTDON *fmt,
               char *format,
               char *datemin, char *datemax );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_FORMAT associée à une donnée de type DATE :

- 'fmt' : adresse de la structure ST_FORMAT à initialiser
- 'format' : format d'une date, du type "date : AA MM JJ"
- 'datemin': date minimale (au format 'format')
- 'datemax': date maximale (au format 'format')

Ce format et ces bornes sont mémorisés dans une structure ST_VERDAT (Cf. Entrées Formatées), créée par allocation dynamique. L'adresse de cette structure est mise dans le pointeur "boîte aux lettres" de la structure ST_FORMAT.

Initialisation des pointeurs sur fonctions :

- la fonction de conversion début **e_cvd_dt()**
- la fonction de conversion fin **e_cvf_dt()**
- la fonction de vérification **e_ver_dt()**
- pas de fonction de recadrage
- le filtre des caractères valides **e_fcv_dt()**
- le filtre des touches spécifiques **e_fts_dt()**

Voir aussi: e_entdat(), e_zondat(), e_cvd_dt(), e_cvf_dt(), e_ver_dt(), e_fcv_dt(), e_fts_dt().

Exemple:
```
ST_FORMAT fmt;

e_fmtdat( &fmt, "Né le : JJ-MM-AAAA",
                "Né le : 01-01-1900", "Né le : 31-12-2000");
```

Nom: e_fmtdbl - initialisation d'une structure ST_FORMAT pour une donnée de type DOUBLE

Usage:
```
void e_fmtflt( ST_ENTDON *fmt,
               int nav, int nap,
               double mini,   double maxi,
               double minasb, double maxabs,
               char *messmin, char *messmax );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_FORMAT associée à une donnée de type DOUBLE :

- 'fmt' : adresse de la structure ST_FORMAT à initialiser
- 'nav' : nombre de chiffres avant le point décimal (signe compris)
- 'nap' : nombre de décimales
- 'mini' : valeur minimale normale (si la valeur saisie est inférieure, on demande confirmation)
- 'maxi' : valeur maximale normale (si la valeur saisie est supérieure, on demande confirmation)
- 'minabs' : valeur minimale absolue (si la valeur saisie est inférieure, on demande de la modifier)
- 'maxabs' : valeur maximale absolue (si la valeur saisie est supérieure, on demande de la modifier)
- 'messmin': message associé au dépassement de la valeur minimale normale (message affiché lors de la demande de confirmation)
- 'messmax': message associé au dépassement de la valeur maximale normale (message affiché lors de la demande de confirmation)

N.B. : la longueur de ces messages doit être inférieure à 35 caractères .

Ces bornes et messages sont mémorisés dans une structure ST_VEREEL (Cf. ENTREES FORMATEES), créée par allocation dynamique. L'adresse de cette structure est mise dans le pointeur "boîte aux lettres" de la structure ST_FORMAT.

Initialisation des pointeurs sur fonctions :

- la fonction de conversion début cadre à gauche **e_cvd_d()**
- la fonction de conversion fin **e_cvf_d()**
- la fonction de vérification **e_ver_d()**
  teste par rapport :
  . aux bornes non bloquantes : mini et maxi,

. aux bornes bloquantes : minabs, maxabs.
- la fonction de recadrage e_cad_r()
  cadre à droite
- le filtre des caractères valides e_fcv_r()
  accepte :
    BLANC, '.', '0' à '9', le signe '+' ou '-' en tête
- le filtre des touches spécifiques e_fts_st
  est la fonction standard

Voir aussi: e_entdbl(), e_zondbl(), e_cvd_d(), e_cvf_d(), e_ver_d(), e_cad_r(), e_fcv_r(), e_fts_st().

Exemple:

```
ST_FORMAT fmt;

 e_fmtdbl( &fmt, 2, 1,
           -50., +50., -60., +80.,
           "Valeur raisonnablement trop faible",
           "Valeur raisonnablement trop élevée");
```

Nom: e_fmtdon - Initialisation d'une structure ST_FORMAT
pour une donnée quelconque

Usage:
```
void e_fmtdon( ST_FORMAT *fmt,
               char *(*f_cvdeb) ( ST_ENTDON *ste ),
               void  (*f_cvfin) ( ST_ENTDON *ste ),
               int   (*f_verif) ( ST_ENTDON *ste ),
               void  (*f_recad) ( ST_ENTDON *ste ),
               int   (*f_carvl) ( int *s, ST_ENTDON *ste ),
               int   (*f_tchsp) ( int *s, ST_ENTDON *ste ),
               void *ptr );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description:
e_fmtdon initialise une structure ST_FORMAT.
'fmt' : adresse de la structure ST_FORMAT à initialiser
'f_cvdeb': pointeur sur la fonction de conversion début (ou NULL)
'f_cvfin': pointeur sur la fonction de conversion fin (ou NULL)
'f_verif': pointeur sur la fonction de vérification (ou NULL)
'f_recad': pointeur sur la fonction de recadrage (ou NULL)
'f_carvl': pointeur sur la fonction "filtre des caractères valides"
'f_tchsp': pointeur sur la fonction "filtre des touches spécifiques"
'ptr' : pointeur "boîte aux lettres", pouvant être utilisé
par chacune des fonctions précédentes
(Cf. ENTREES FORMATEES)

Voir aussi: e_fmt???(), e_cvd_?(), e_cvf_?(), e_ver_?(), e_cad_?(),
e_fcv_?(), e_fts_?(),

Exemple:
```
ST_FORMAT fmt;

 e_fmtdon( &fmt, NULL, NULL, NULL, e_cad_a, e_fcv_st, e_fts_st, NULL);

 /* Initialise un format de type standard 'chaîne de caractères' :
    - pas de fct de conversion
    - pas de fct de vérification
    - recadrage à gauche
    - 'filtre des caractères valides' standard
    - 'filtre des touches spécifiques' standard    */
```

Nom: e_fmtflt - initialisation d'une structure ST_FORMAT pour une donnée de type FLOAT

Usage:
```
void e_fmtflt( ST_ENTDON *fmt,
               int nav, int nap,
               float mini,   float maxi,
               float minasb, float maxabs,
               char *messmin, char *messmax );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_FORMAT associée à une donnée de type FLOAT :

- 'fmt' : adresse de la structure ST_FORMAT à initialiser
- 'nav' : nombre de chiffres avant le point décimal (signe compris)
- 'nap' : nombre de décimales
- 'mini' : valeur minimale normale (si la valeur saisie est inférieure, on demande confirmation)
- 'maxi' : valeur maximale normale (si la valeur saisie est supérieure, on demande confirmation)
- 'minabs' : valeur minimale absolue (si la valeur saisie est inférieure, on demande de la modifier)
- 'maxabs' : valeur maximale absolue (si la valeur saisie est supérieure, on demande de la modifier)
- 'messmin': message associé au dépassement de la valeur minimale normale (message affiché lors de la demande de confirmation)
- 'messmax': message associé au dépassement de la valeur maximale normale (message affiché lors de la demande de confirmation)

N.B. : la longueur de ces messages doit être inférieure à 35 caractères .

Ces bornes et messages sont mémorisés dans une structure ST_VEREEL (Cf. Saisie d'un entier), créée par allocation dynamique. L'adresse de cette structure est mise dans le pointeur "boîte aux lettres" de la structure ST_FORMAT.

Initialisation des pointeurs sur fonctions :

- la fonction de conversion début cadre à gauche : e_cvd_f()
- la fonction de conversion fin : e_cvf_f()
- la fonction de vérification : e_ver_f()
  teste par rapport :
  . aux bornes non bloquantes : mini et maxi,
  . aux bornes bloquantes : minabs, maxabs.

- la fonction de recadrage e_cad_r()
  cadre à droite
- le filtre des caractères valides e_fcv_r()
  accepte :
    BLANC, '.', '0' à '9', le signe '+' ou '-' en tête
- le filtre des touches spécifiques e_fts_st
  est la fonction standard

Voir aussi: e_entflt(), e_zonflt(), e_cvd_f(), e_cvf_f(), e_ver_f(), e_cad_r(), e_fcv_r(), e_fts_st().

Exemple:

```
ST_FORMAT fmt;

e_fmtflt( &fmt, 2, 1, -50., +50., -60., +80.,
          "Valeur raisonnablement trop faible",
          "Valeur raisonnablement trop élevée");
```

Nom: e_fmtint - initialisation d'une structure ST_FORMAT pour une donnée de type INT

Usage:

```
void e_fmtint( ST_ENTDON *fmt,
               int mini,   int maxi,
               int minasb, int maxabs,
               char *messmin, char *messmax );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_FORMAT associée à une donnée de type INT :

- 'fmt' : adresse de la structure ST_FORMAT à initialiser
- 'mini' : valeur minimale normale (si la valeur saisie est inférieure, on demande confirmation)
- 'maxi' : valeur maximale normale (si la valeur saisie est supérieure, on demande confirmation)
- 'minabs' : valeur minimale absolue (si la valeur saisie est inférieure, on demande de la modifier)
- 'maxabs' : valeur maximale absolue (si la valeur saisie est supérieure, on demande de la modifier)
- 'messmin': message associé au dépassement de la valeur minimale normale (message affiché lors de la demande de confirmation)
- 'messmax': message associé au dépassement de la valeur maximale normale (message affiché lors de la demande de confirmation)

N.B. : la longueur de ces messages doit être inférieure à 35 caractères .

Ces bornes et messages sont mémorisés dans une structure ST_VERENT (Cf. Saisie d'un entier), créée par allocation dynamique. L'adresse de cette structure est mise dans le pointeur "boîte aux lettres" de la structure ST_FORMAT.

Initialisation des pointeurs sur fonctions :

- la fonction de conversion début cadre à gauche : e_cvd_i()
- la fonction de conversion fin : e_cvf_i()
- la fonction de vérification : e_ver_i()
  teste par rapport :
  . aux valeurs min et max d'un int (-32767, 32767)
  . aux bornes non bloquantes : mini et maxi,
  . aux bornes bloquantes : minabs, maxabs.

- la fonction de recadrage e_cad_e()
  cadre à droite
- le filtre des caractères valides e_fcv_e()
  accepte :
    BLANC, '0' à '9', le signe '+' ou '-' en tête
- le filtre des touches spécifiques e_fts_st()
  est la fonction standard

Voir aussi: e_entint(), e_zonint(), e_cvd_i(), e_cvf_i(), e_ver_i(), e_cad_e(), e_fcv_e(), e_fts_st().

Exemple:

```
ST_FORMAT fmt;

 e_fmtint( &fmt, -100, +100, -150, 200,
           "Valeur raisonnablement trop faible",
           "Valeur raisonnablement trop élevée");
```

Nom: e_fmtlng - initialisation d'une structure ST_FORMAT pour une donnée de type LONG INT

Usage:
```
void e_fmtlng( ST_ENTDON *fmt,
               long mini,   long maxi,
               long minasb, long maxabs,
               char *messmin, char *messmax );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_FORMAT associée à une donnée de type LONG INT :

- 'fmt' : adresse de la structure ST_FORMAT à initialiser
- 'mini' : valeur minimale normale (si la valeur saisie est inférieure, on demande confirmation)
- 'maxi' : valeur maximale normale (si la valeur saisie est supérieure, on demande confirmation)
- 'minabs' : valeur minimale absolue (si la valeur saisie est inférieure, on demande de la modifier)
- 'maxabs' : valeur maximale absolue (si la valeur saisie est supérieure, on demande de la modifier)
- 'messmin': message associé au dépassement de la valeur minimale normale (message affiché lors de la demande de confirmation)
- 'messmax': message associé au dépassement de la valeur maximale normale (message affiché lors de la demande de confirmation)

N.B. : la longueur de ces messages doit être inférieure à 35 caractères .

Ces bornes et messages sont mémorisés dans une structure ST_VERENT (Cf. Saisie d'un entier), créée par allocation dynamique. L'adresse de cette structure est mise dans le pointeur "boîte aux lettres" de la structure ST_FORMAT.

Initialisation des pointeurs sur fonctions :

- la fonction de conversion début cadre à gauche — e_cvd_l()
- la fonction de conversion fin — e_cvf_l()
- la fonction de vérification — e_ver_l()
  teste par rapport :
  . aux bornes non bloquantes : mini et maxi,
  . aux bornes bloquantes : minabs, maxabs.
- la fonction de recadrage cadre à droite — e_cad_e()
- le filtre des caractères valides — e_fcv_e()

accepte :
BLANC, '0' à '9', le signe '+' ou '-' en tête
- le filtre des touches spécifiques **e_fts_st**
est la fonction standard

Voir aussi: e_entlng(), e_zonlng(), e_cvd_l(), e_cvf_l(), e_ver_l(), e_cad_e(), e_fcv_e(), e_fts_st().

Exemple:

```
ST_FORMAT fmt;

 e_fmtint( &fmt, -50000L, +100000L, -1000000L, +2000000L,
           "Valeur raisonnablement trop faible",
           "Valeur raisonnablement trop élevée");
```

Nom: e_fmtuns - initialisation d'une structure ST_FORMAT pour une donnée de type UNSIGNED

Usage:
```
void e_fmtuns( ST_ENTDON *fmt,
               unsigned mini,   unsigned maxi,
               unsigned minasb, unsigned maxabs,
               char *messmin, char *messmax );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_FORMAT associée à une donnée de type UNSIGNED :

- 'fmt' : adresse de la structure ST_FORMAT à initialiser
- 'mini' : valeur minimale normale (si la valeur saisie est inférieure, on demande confirmation)
- 'maxi' : valeur maximale normale (si la valeur saisie est supérieure, on demande confirmation)
- 'minabs' : valeur minimale absolue (si la valeur saisie est inférieure, on demande de la modifier)
- 'maxabs' : valeur maximale absolue (si la valeur saisie est supérieure, on demande de la modifier)
- 'messmin': message associé au dépassement de la valeur minimale normale (message affiché lors de la demande de confirmation)
- 'messmax': message associé au dépassement de la valeur maximale normale (message affiché lors de la demande de confirmation)

N.B. : la longueur de ces messages doit être inférieure à 35 caractères .

Ces bornes et messages sont mémorisés dans une structure ST_VERENT (Cf. Saisie d'un entier), créée par allocation dynamique. L'adresse de cette structure est mise dans le pointeur "boîte aux lettres" de la structure ST_FORMAT.

Initialisation des pointeurs sur fonctions :

- la fonction de conversion début cadre à gauche **e_cvd_u()**
- la fonction de conversion fin **e_cvf_u()**
- la fonction de vérification **e_ver_u()**
  teste par rapport :
  . aux valeurs min et max d'un unsigned (0, 65535)
  . aux bornes non bloquantes : mini et maxi,
  . aux bornes bloquantes : minabs, maxabs.
- la fonction de recadrage cadre à droite **e_cad_e()**

- le filtre des caractères valides e_fcv_e()
  accepte :
    BLANC, '0' à '9', le signe '+' ou '-' en tête
- le filtre des touches spécifiques e_fts_st()
  est la fonction standard

Voir aussi: e_entuns(), e_zonuns(), e_cvd_u(), e_cvf_u(), e_ver_u(), e_cad_e(), e_fcv_e(), e_fts_st().

Exemple:

```
ST_FORMAT fmt;

 e_fmtuns( &fmt, 10U, 100U, 2U, 200U,
           "Valeur raisonnablement trop faible",
           "Valeur raisonnablement trop élevée");
```

Nom: e_pargen - initialisation d'une structure ST_PARGEN.

Usage:

```
void  e_pargen( ST_PARGEN *parec,
                int sortg, int sortd, int dep,
                int car,
                int tmod, int tinit, int teff ) ;
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description:

e_pargen initialise une structure ST_PARGEN.

```
parec : adresse de la structure ST_PARGEN à initialiser
sortg : sortie par la gauche autorisée ( OUI ou NON )
sortd : sortie par la droite autorisée ( OUI ou NON )
dep   : départ automatique autorisé    ( OUI ou NON )
car   : caractère de remplissage       ( >31 et <255 )
tmod  : touche d'entrée dans la zone   ( code ou 0 )
tinit : touche de réinitialisation de zone  ( id )
teff  : touche d'effacement                 ( id )
```

Voir aussi: e_entdon(), e_fentre().

Exemple:

```
ST_PARGEN parec;

e_pargen( parec, OUI, OUI, NON, '.', 0, 0, 0);
```

Nom: e_ver_d - Fonction de Vérification pour une donnée de type DOUBLE

Usage: int e_ver_d(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- OUI : la valeur saisie est correcte
- NON : la valeur saisie est incorrecte

Description: Vérification de la valeur saisie par rapport aux bornes définies dans la structure ST_VEREEL associée (ste->Ptr) S'il n'y en a pas, la fonction retourne OUI

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Voir aussi: e_fmtdbl()

Nom: e_ver_dt - Fonction de Vérification pour une donnée de type DATE

Usage: int e_ver_dt(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- OUI : la valeur saisie est correcte
- NON : la valeur saisie est incorrecte

Description: Vérification de la valeur saisie
1) par rapport aux bornes générales d'une DATE (jour, mois, et problème des années bissextiles)
2) par rapport aux bornes définie dans la structure ST_VERDAT associée (ste->Ptr). S'il n'y en a pas, la fonction retourne OUI

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Voir aussi: e_fmtdat()

Nom: e_ver_f - Fonction de Vérification pour une donnée de type FLOAT

Usage: int e_ver_f(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- OUI : la valeur saisie est correcte
- NON : la valeur saisie est incorrecte

Description: Vérification de la valeur saisie par rapport aux bornes définies dans la structure ST_VEREEL associée (ste->Ptr) S'il n'y en a pas, la fonction retourne OUI

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Voir aussi: e_fmtdbl()

Nom: e_ver_i - Fonction de Vérification pour une donnée de type INT

Usage: int e_ver_i(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- OUI : la valeur saisie est correcte
- NON : la valeur saisie est incorrecte

Description: Vérification de la valeur saisie
1) par rapport aux bornes d'un INT
2) par rapport aux bornes définie dans la structure ST_VERENT associée (ste->Ptr). S'il n'y en a pas, la fonction retourne OUI

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Voir aussi: e_fmtint()

Nom: e_ver_l - Fonction de Vérification pour une donnée de type LONG

Usage: int e_ver_l(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- OUI : la valeur saisie est correcte
- NON : la valeur saisie est incorrecte

Description: Vérification de la valeur saisie par rapport aux bornes définie dans la structure ST_VERENT associée (ste->Ptr). S'il n'y en a pas, la fonction retourne OUI

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Voir aussi: e_fmtlng()

Nom: e_ver_u - Fonction de Vérification pour une donnée de type UNSIGNED

Usage: int e_ver_u(ST_ENTDON *ste);

Prototype dans: grentre.h

Valeur retournée:
- OUI : la valeur saisie est correcte
- NON : la valeur saisie est incorrecte

Description: Vérification de la valeur saisie
1) par rapport aux bornes d'un UNSIGNED
2) par rapport aux bornes définie dans la structure ST_VERENT associée (ste->Ptr). S'il n'y en a pas, la fonction retourne OUI

- 'ste' : structure ST_ENTDON associée à la donnée à saisir

Voir aussi: e_fmtuns()

Nom: e_zonalp - initialisation d'une structure ST_ENTDON
pour la saisie d'une chaîne de caractères

Usage:
```
void e_zonalp( ST_ENTDON *ste,
               ST_FORMAT *fmt,
               int ecran, int lig, int col,
               char *donnee, int lg );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_ENTDON repérée par son adresse 'ste' :

- 'ste' : adresse de la structure ST_ENTDON à initialiser
- 'fmt' : adresse de la structure ST_FORMAT associée au type de donnée saisie .L'envoi d'un pointeur NULL entraîne la création d'une structure ST_FORMAT standard, statique,

  propre aux données de type chaîne de caractères où tous les pointeurs sont NULL, fmt->Cad qui pointe sur la fonction e_cad_a(), qui opère un recadrage à gauche pour une chaîne de caractère.
- 'ecran' : n° d'écran virtuel, ou n° de palette en écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir (ne pas oublier d'initialiser

  cette donnée avant de la saisir !)
- 'lg' : longueur de la zone de saisie (= longueur de la chaîne)

Voir aussi: e_entalp(), e_fmtalp(), e_cad_a()

Exemple:
```
ST_ENTDON ste;
char chaine[21];
int palette;

palette = p_crepal( 0, 112, 127, 125, 0, 0);
e_zonalp( &ste, NULL, palette, 10, 5, chaine, 20);
```

Nom: e_zoncar - initialisation d'une structure ST_ENTDON
pour la saisie d'une chaîne de caractères

Usage:
```
void e_zoncar( ST_ENTDON *ste,
               ST_FORMAT *fmt,
               int ecran, int lig, int col,
               char *donnee);
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la la structure ST_ENTDON repérée par son adresse 'ste' :

- 'ste' : adresse de la structure ST_ENTDON à initialiser
- 'fmt' : adresse de la structure ST_FORMAT associée au type de donnée saisie .L'envoi d'un pointeur NULL entraîne la création d'une structure ST_FORMAT standard, statique,

  propre aux données de type caractère
  où tous les pointeurs sont NULL, sauf fmt->Cvd et fmt->Cvf qui permettent les conversions début et fin.
- 'ecran' : n° d'écran virtuel, ou n° de palette en écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir (ne pas oublier d'initialiser

  cette donnée avant de la saisir !)

Voir aussi: e_entcar(), e_fmtcar(), e_cvd_c(), e_cvf_c()

Exemple:
```
ST_ENTDON ste;
char car;
int palette;

palette = p_crepal( 0, 112, 127, 125, 0, 0);
e_zoncar( &ste, NULL, palette, 10, 5, &car);
```

Nom: e_zondat - initialisation d'une structure ST_ENTDON
pour la saisie d'une donnée de type DATE

Usage:
```
void e_zondat( ST_ENTDON *ste,
               ST_FORMAT *fmt,
               int ecran, int lig, int col,
               char *donnee);
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_ENTDON repérée par son adresse 'ste' :

- 'ste' : adresse de la structure ST_ENTDON à initialiser
- 'fmt' : adresse de la structure ST_FORMAT associée au type de donnée saisie.
  Si fmt == NULL : création d'une structure ST_FORMAT standard, statique, propre aux données de type DATE, avec le format implicite "JJ/MM/AA", sans borne (pour plus de précision Cf. e_fmtdat())
- 'ecran' : n° d'écran virtuel, ou n° de palette en écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la date à saisir (ne pas oublier d'initialiser la chaîne avant de la saisir !)

Voir aussi: e_entdat(), e_fmtdat(), e_cvd_dt(), e_cvf_dt(), e_ver_dt(), e_fcv_dt(), e_fts_dt()

Exemple:
```
ST_ENTDON ste;
ST_FORMAT fmt;
char date[21]
int palette;

palette = p_crepal( 0, 112, 127, 125, 0, 0);
e_fmtdat( &fmt, "Né le : JJ-MM-AAAA",
                "Né le : 01-01-1900", "Né le : 31-12-2000");
e_zondat( &ste, &fmt, palette, 10, 40, date);
```

Nom: e_zondbl - initialisation d'une structure ST_ENTDON
pour la saisie d'une donnée de type DOUBLE

Usage:
```
void e_zondbl( ST_ENTDON *ste,
               ST_FORMAT *fmt,
               int ecran, int lig, int col,
               double *donnee, int lg );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_ENTDON repérée par son adresse 'ste' :

- 'ste' : adresse de la structure ST_ENTDON à initialiser
- 'fmt' : adresse de la structure ST_FORMAT associée au type de donnée saisie.
  Si fmt == NULL : création d'une structure ST_FORMAT standard, statique, propre aux données de type DOUBLE, (pour plus de précision Cf. e_fmtdbl())
- 'ecran' : n° d'écran virtuel, ou n° de palette en écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir (ne pas oublier d'initialiser cette donnée avant de la saisir !)
- 'lg' : longueur de la zone de saisie

Voir aussi: e_entdbl(), e_fmtdbl(), e_cvd_d(), e_cvf_d(), e_ver_d(), e_cad_r(), e_fcv_r(), e_fts_st()

Exemple:
```
ST_ENTDON ste;
ST_FORMAT fmt;
double reel;
int palette;

palette = p_crepal( 0, 112, 127, 125, 0, 0);
e_fmtdbl( &fmt, 0., 0., 0., 20., NULL, NULL);
e_zondbl( &ste, &fmt, palette, 10, 40, reel, 3);
```

Nom: e_zonflt - initialisation d'une structure ST_ENTDON pour la saisie d'une donnée de type FLOAT

Usage:

```
void e_zonflt( ST_ENTDON *ste,
               ST_FORMAT *fmt,
               int ecran, int lig, int col,
               float *donnee, int lg );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_ENTDON repérée par son adresse 'ste' :

- 'ste' : adresse de la structure ST_ENTDON à initialiser
- 'fmt' : adresse de la structure ST_FORMAT associée au type de donnée saisie.
  Si fmt == NULL : création d'une structure ST_FORMAT standard, statique, propre aux données de type FLOAT, mais sans autre vérification que celle liée au type FLOAT (pour plus de précision Cf. e_fmtflt())
- 'ecran' : n° d'écran virtuel, ou n° de palette en écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir
- 'lg' : longueur de la zone de saisie (lg <= 8)

Voir aussi: e_entflt(), e_fmtflt(), e_cvd_f(), e_cvf_f(), e_ver_f(), e_cad_r(), e_fcv_r(), e_fts_st()

Exemple:

```
ST_ENTDON ste;
ST_FORMAT fmt;
float reel;
int palette;

palette = p_crepal( 0, 112, 127, 125, 0, 0);
e_fmtflt( &fmt, 0., 0., 0., 20., NULL, NULL);
e_zonflt( &ste, &fmt, palette, 10, 40, reel, 3);
```

Nom: e_zonint - initialisation d'une structure ST_ENTDON pour la saisie d'une donnée de type INT

Usage:
```
void e_zonint( ST_ENTDON *ste,
               ST_FORMAT *fmt,
               int ecran, int lig, int col,
               int *donnee, int lg );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_ENTDON repérée par son adresse 'ste' :

- 'ste' : adresse de la structure ST_ENTDON à initialiser
- 'fmt' : adresse de la structure ST_FORMAT associée au type de donnée saisie.
  Si fmt == NULL : création d'une structure ST_FORMAT standard, statique, propre aux données de type INT, mais sans vérification autre que celle correspondant au type INT (pour plus de précision Cf. e_fmtint())
- 'ecran' : n° d'écran virtuel, ou n° de palette en écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir
- 'lg' : longueur de la zone de saisie (lg <= 5)

Voir aussi: e_entint(), e_fmtint(), e_cvd_i(), e_cvf_i(), e_ver_i(), e_cad_e(), e_fcv_e(), e_fts_st()

Exemple:
```
ST_ENTDON ste;
ST_FORMAT fmt;
int note
int palette;

palette = p_crepal( 0, 112, 127, 125, 0, 0);
e_fmtint( &fmt, 0, 0, 0, 20, NULL, NULL);
e_zonint( &ste, &fmt, palette, 10, 40, note, 2);
```

Nom: e_zonlng - initialisation d'une structure ST_ENTDON pour la saisie d'une donnée de type LONG

Usage:
```
void e_zonlng( ST_ENTDON *ste,
               ST_FORMAT *fmt,
               int ecran, int lig, int col,
               long *donnee, int lg );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_ENTDON repérée par son adresse 'ste' :

- 'ste' : adresse de la structure ST_ENTDON à initialiser
- 'fmt' : adresse de la structure ST_FORMAT associée au type de donnée saisie.
  Si fmt == NULL : création d'une structure ST_FORMAT standard, statique, propre aux données de type LONG, mais sans vérification autre que celle correspondant au type LONG (pour plus de précision Cf. e_fmtlng())
- 'ecran' : n° d'écran virtuel, ou n° de palette en écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir
- 'lg' : longueur de la zone de saisie (lg <= 10)

Voir aussi: e_entlng(), e_fmtlng(), e_cvd_l(), e_cvf_l(), e_ver_l(), e_cad_e(), e_fcv_e(), e_fts_st()

Exemple:
```
ST_ENTDON ste;
ST_FORMAT fmt;
long note;
int palette;

palette = p_crepal( 0, 112, 127, 125, 0, 0);
e_fmtlng( &fmt, 0, 0, 0, 20, NULL, NULL);
e_zonlng( &ste, &fmt, palette, 10, 40, note, 2);
```

Nom: e_zonuns - initialisation d'une structure ST_ENTDON
pour la saisie d'une donnée de type UNSIGNED

Usage:
```
void e_zonuns( ST_ENTDON *ste,
               ST_FORMAT *fmt,
               int ecran, int lig, int col,
               unsigned *donnee, int lg );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation de la structure ST_ENTDON repérée par son adresse 'ste' :

- 'ste' : adresse de la structure ST_ENTDON à initialiser
- 'fmt' : adresse de la structure ST_FORMAT associée au type de donnée saisie.
  Si fmt == NULL : création d'une structure ST_FORMAT standard, statique, propre aux données de type UNSIGNED, mais sans vérification autre que celle correspondant au type UNSIGNED (pour plus de précision Cf. e_fmtuns())
- 'ecran' : n° d'écran virtuel, ou n° de palette en écran physique
- 'lig'
- 'col' : coordonnées du début de la zone de saisie dans l'écran
- 'donnee' : adresse de la donnée à saisir
- 'lg' : longueur de la zone de saisie (lg <= 5)

Voir aussi: e_entuns(), e_fmtuns(), e_cvd_u(), e_cvf_u(), e_ver_u(), e_cad_e(), e_fcv_e(), e_fts_st()

Exemple:
```
ST_ENTDON ste;
ST_FORMAT fmt;
unsigned note;
int palette;

palette = p_crepal( 0, 112, 127, 125, 0, 0);
e_fmtuns( &fmt, 0, 0, 0, 20, NULL, NULL);
e_zonuns( &ste, &fmt, palette, 10, 40, note, 2);
```

Nom: e_dfsais - Initialisation d'une structure ST_SAISIE
associée à une zone de saisie dans une FICHE

Usage:

```
void e_dfsais(
                ST_SAISIE *sts,
                int ecran, int lig, int col,
                void *donnee,
                int lg,
                ST_FORMAT *fmt,
                int (*saisie)( ST_PARGEN *, ST_ENTDON *, int ),
                int haut, int bas, int droit, int gauche
                );
```

Prototype dans: grentre.h

Valeur retournée: aucune

Description: Initialisation d'une structure ST_SAISIE :

- 'sts' : adresse de la structure ST_SAISIE à instancier

permettent d'instancier 'sts->Ste'

- 'ecran' : n° d'ecran VSI, ou n° de palette
- 'lig' :
- 'col' : coordonnées du début de la zone de saisie
- 'donnee' : adresse de la donnée à saisir
- 'lg' : longueur de la zone de saisie
- 'fmt' : adresse de la structure ST_FORMAT associée (ou NULL)

- 'saisie' : pointeur sur la fonction personnelle de saisie (ou NULL). Cette fonction doit avoir pour arguments :
  . l'adresse de la structure ST_PARGEN contenant les paramètres généraux de saisie
  . l'adresse de la structure ST_ENTDON associée à la donnée à saisir (contenue dans 'sts')
  . le numéro de la zone à saisir

- 'haut' :
- 'bas' :
- 'droit' :
- 'gauche' : gestion des déplacements de zone à zone dans la fiche : indice de la "zone" dans le tableau de structure ST_SAISIE constituant le "plan de saisie" de la fiche (-1 pour sortir)

Voir aussi: e_saisie()

Exemple: Cf. listing du programme **testsais.c**

Nom: e_saisie - Saisie d'une "fiche" constituée de plusieurs données à saisir, avec gestion automatique des déplacements de zone à zone

Usage:

```
int  e_saisie(
                ST_PARGEN *parec,
                ST_SAISIE plan[],
                int *numero,
                int *modif
                );
```

Prototype dans: grentre.h

Valeur retournée: code de la touche de sortie

Description:

- 'parec' : adresse de la structure ST_PARGEN contenant les paramètres généraux de saisie. Si 'parec' == NULL e_saisie utilise une structure implicite, statique (Cf. e_ent???())

- 'plan' : adresse du tableau de structure ST_SAISIE constituant le "plan de saisie" de la fiche

- 'numero' : adresse du n° de la première zone à saisir dans la fiche. Au retour, '*numero' est actualisé au n° de la dernière zone saisie

- 'modif' : drapeau de modification "globale" de la fiche. Au retour '*modif' = OUI si au moins l'une des zones a été modifiée

Fonctionnement :

- Si parec==NULL, définition d'une structure ST_PARGEN implicite statique (aucune sortie autorisée, remplissage par des blancs, aucune touche spéciale)

- Indice de la zone à saisir : i = *numero

- Si utilisation d'écrans virtuels, mémorisation de l'écran en cours, et ouverture de la fenêtre s'il y a changement d'écran virtuel (=>on peut donc saisir des zones situées dans des fenêtres différentes).

- S'il y a un pointeur sur une fonction de saisie personnelle (plan[i].Saisie != NULL), appel à cette fonction; sinon appel à e_entdon().
  L'une et l'autre actualisent le drapeau de modification dans la structure ST_ENTDON associée à la zone saisie (plan[i].Ste.Modif), et renvoient la touche de sortie de la zone

- Actualisation du drapeau global de modification ('*modif')

- Actualisation du n° de la dernière zone saisie ('*numero')

- Traitement de la touche de sortie d'une zone:
. HAUT: indice de la nouvelle zone à saisir = plan_saisie[i].Haut
. BAS : indice de la nouvelle zone à saisir = plan_saisie[i].Bas
. C_DROIT , DROIT: ..... = plan_saisie[i].Droit
. C_GAUCHE , GAUCHE: ..... = plan_saisie[i].Gauche

Si indice < 0 ou autre touche, fin de la saisie et on renvoie le code de la touche de sortie.

Voir aussi: e_dfsais()

Exemple: Cf. listing du programme **testsais.c**

Nom: pkl_aff - Affichage d'une "picklist"

Usage: void pkl_aff(ST_PKL *pkl, int num0, char cadre);

Prototype dans: grpkl.h

Valeur retournée: aucune

Description: Affiche la "picklist" 'pkl' à partir de l'élément 'num0'
avec son cadre si 'cadre' = OUI
Cette fonction est utilisée par pkl_fct()

Nom: pkl_cat - Concatène deux "picklist"

Usage: ST_PKL *pkl_cat(ST_PKL *pkl1, ST_PKL *pkl2);

Prototype dans: grpkl.h

Valeur retournée: adresse de la "picklist" obtenue (= 'pkl1')

Description: Recopie la liste de 'pkl2' à la fin de celle de 'pkl1'.
La liste de 'pkl1' ('pkl1->Liste') est éventuellement réallouée.
Retourne 'pkl1'.

Voir aussi: pkl_pkl()

Nom: pkl_cre - Création et initialisation d'une "picklist"

Usage:

```
ST_PKL *pkl_cre ( char *titre,
                  void **liste,
                  char *(*nom)(void *),
                  int nbelt, int nbmax,
                  int lchg, int cchg,
                  int nlig, int ncol, int ncar,
                  int palette,
                  char ajout, char tri,
                  int (*fct)(ST_PKL *) );
```

Prototype dans: grpkl.h

Valeur retournée: adresse de la structure ST_PKL créée et instanciée

Description: pkl_cre() crée une structure ST_PKL par allocation dynamique, l'initialise avec les arguments spécifiée, et renvoie son pointeur.

- 'titre' : titre de la "picklist", qui sera complété par des '-' en tête et en queue dans 'pkl->Titre' alloué dynamiquement
- 'liste' : adresse de la liste. Cette liste doit être un vecteur de pointeurs, chaque pointeur pointant sur un élément de la liste.
- 'nom' : fonction renvoyant le nom d'1 élément de la liste

  exemple :

```
char *nom_elt( void *elt )
{
struct truc
       {
       char nom[21];
       ...
       } *ptr;

ptr = (struct truc *) elt;
                /* ne pas oublier le "casting" ! */

return( ptr->nom );
}
```

  N.B. - le pointeur NULL est remplacé par la fonction implicite pkl_nom()

- 'nbelt' : nombre (actuel) d'éléments dans la liste
- 'nbmax' : taille (actuelle) de la liste (= taille du vecteur de pointeur)
- 'lchg'

- 'cchg' : coordonnees du coin haut gauche du cadre
  de la pickliste (s'il y en a un)
- 'nlig'
- 'ncol' : nombre de lignes et de colonnes dans la fenêtre
  de la "picklist"
- 'ncar' : nombre de caractères par colonne
- 'palette' : palette de couleurs (cf. p_crepal);
  seuls a1 (Vlcadre), a2 (Vlnorm) et a3 (Vlmodif)
  sont utilisés;
  si a1 = 0, la pickliste n'aura ni titre, ni cadre
  (=> lchg et cchg sont les coodonnées du coin haut
  gauche de la fenêtre "picklist" sans cadre).
- 'ajout' : drapeau d'ajout possible (OUI/NON)
- 'tri' : drapeau de tri alphabétique (OUI/NON)
- 'fct' : fonction utilisateur associée à pkl_fct();
  elle a pour argument le pointeur sur la "picklist"
  en cours et renvoie un code (Cf. pkl_fct())

Voir aussi: pkl_???()

Exemple:

```
ST_PKL *pkl;
struct truc
       {
       char nom[21];
       ...
       } **liste;
char *nom_elt( void *elt );
int palette;

palette = p_crepal( 31, 127, 95, 0, 0, 0);

pkl = pkl_cre( " PICKLIST ", (void **) liste, nom_elt,
               0, 0, /* liste vide, taille = 0 */
               5, 30,
               10, 1, 20;
               palette;
               OUI, OUI, NULL);
```

Nom: pkl_eff - Efface une "picklist" et remet l'écran dans l'état initial si on a mémorisé la zone initiale

Usage: void pkl_eff(ST_PKL *pkl);

Prototype dans: grpkl.h

Valeur retournée: aucune

Description: Efface la "picklist" 'pkl', et remet l'écran dans l'état initial si on a mémorisé auparavant la zone initiale grâce à pkl_zon().
S'utilise géneralement après pkl_fct()

Nom: pkl_fct - fonction PICKLIST

Usage: int pkl_fct(ST_PKL *pkl);

Prototype dans: grpkl.h

Valeur retournée: numéro de l'élément choisi (numéro = nb_elt si [...] sélectionné) ou -1 si sortie sans sélection

Description: Affiche la "picklist" spécifiée, et gère les déplacements pour sélectionner un élément dans la liste. La sélection d'un élément se fait par la touche ENTER; la sortie sans sélection se fait par la touche ESCAPE.

- 'pkl' : adresse de la structure ST_PKL

Fonctionnement :

- Affichage de la "picklist"
- Pointage sur l'éventuel élément précédemment pointé
- TANT QUE la sortie n'est pas demandée

  . appel à la fonction associée (si 'pkl->Fct' != NULL)
  . attente d'un caractère au clavier (si code de retour de 'pkl->Fct' < 0)
  . gestion des déplacements :
  ENTER : sélection de l'élément et sortie de boucle
  ESCAPE : sortie sans sélection (numéro = -1)
  DROIT - GAUCHE - BAS - HAUT : déplacements classiques
  HOME - END : début et fin de page
  C_PRECEDE - C_HOME : début de liste
  C_SUITE - C_END : fin de liste
  caractère quelconque : recherche du premier élément suivant commençant par ce caractère

- RETOUR : numéro de l'élément sélectionné

N.B. - l'élément pointé est affiché en couleur a2 (Vlmodif) et son numéro est mémorisé dans pkl->Elt.
- si pkl->Fct!=NULL, celle-ci est appelée chaque fois (ce qui permet notamment de donner des précisions sur l'élément pointé). Si le code retourné est >= 0, il est directement traité dans la boucle (pas de enclav()).
- le curseur est invisible dans la fonction; il est restitué à la sortie
- avant l'appel de pkl_fct() il est conseillé de faire pkl_zon() !

Voir aussi: pkl_???()

Exemple: Cf. programme d'exemple **testpkl.c**

Nom: pkl_lib - Destruction d'une "picklist"

Usage: void pkl_lib(ST_PKL *pkl);

Prototype dans: grpkl.h

Valeur retournée: aucune

Description: Libere la place allouée pour la "picklist" 'pkl', mais pas la liste.

Nom: pkl_nom - Fonction implicite renvoyant le nom d'un élément d'une liste de chaine de caractères

Usage: char *pkl_nom(void *elt);

Prototype dans: grpkl.h

Valeur retournée: nom de l'élément de la "picklist"

Description: Utilisée par les fonctions PICKLIST lorsqu'à la création de la "picklist" l'utilisateur a envoyé un pointeur NULL (Cf. pkl_cre())

- 'elt' : adresse de l'élément de la "picklist"

```
char *pkl_nom( void *elt)
{
return ((char *) elt);
}
```

Nom: pkl_pkl - Remplit une "picklist" à partir d'une "picklist" source

Usage: void **pkl_pkl(ST_PKL *destin, ST_PKL *source);

Prototype dans: grpkl.h

Valeur retournée: adresse de la liste destination ('destin->Liste')

Description:
- 'destin' : pointeur sur la "picklist' destination
- 'source' : pointeur sur la "picklist' source

La liste destination ('destin->Liste') est construite à partir d'éléments sélectionnés dans la "picklist" 'source'.

La "picklist" destination est affichée et positionnée sur [...] (repère d'ajout); sa sélection entraine l'affichage de la "picklist" source (sans ajout).

Les éléments choisis dans la source sont insérés dans la liste destination (insertion dichotomique si 'destin' triée ('destin->Tri' = OUI), chronologique sinon). Pour revenir dans la "picklist" 'destin', tapez ESCAPE.

La sélection d'un élément dans 'destin' entraine sa suppression.

Quand la construction est terminée, tapez ESCAPE.

N.B. : - la zone d'affichage de la source est mémorisée automatiquement; ce qui n'est pas le cas pour la pickliste destination. Donc, à vous de jouer avec pkl_zon() et pkl_eff() !
- Ne pas oublier de réactualiser le pointeur sur sa liste destination car 'destin->Liste' a pu être réalloué par pkl_pkl() !

Exemple: Cf. programme d'exemple **testpkl.c**

Nom: pkl_rch - Recherche (insertion/suppression) d'un élément dans une "picklist"

Usage:
```
int pkl_rch( ST_PKL *pkl, void *elt,
                    char *nouveau, int option );
```

Prototype dans: grpkl.h

Valeur retournée: position de l'élément dans la liste

Description:

- 'pkl' : pointeur sur la "picklist"
- 'elt' : élément à rechercher/insérer/supprimer dans la liste (de même type que les autres éléments de la liste
- 'nouveau' : drapeau (au retour '*nouveau' = OUI si l'élément n'est pas déja dans la liste, '*nouveau' = NON sinon)
- 'option' : option de recherche
  - -1 -> supprime l'élément dans la liste (s'il n'est pas nouveau)
  - 1 -> insere l'élément dans la liste (s'il est nouveau)
  - 0 -> ne donne que la position de l'élément dans la liste

Recherche dichotomique ou systématique (suivant que la liste est triée ('pkl->Tri' = OUI) ou non) de l'élément dans la liste.

. Si option = 1 et l'élément est nouveau, elle l'insere (insertion dichotomique si liste triée, en fin de liste sinon) en augmentant la taille de la liste si nécessaire, et renvoie sa position dans la nouvelle liste.

. Si option = -1 et l'élément est déja dans la liste, elle le supprime et renvoie sa position dans l'ancienne liste.

Dans les deux cas, la liste est modifiée et le nombre d'élément de la liste est actualisé.

. Si option = 0, la valeur de retour est la position de l'élément dans la liste.

ATTENTION : Ne pas oublier de réactualiser le pointeur sur sa liste si 'option' = 1 et 'nouveau' = OUI, car 'pkl->Liste' a pu être réalloué !

Exemple:

```
ST_PKL *pkl;
struct truc
       {
       char *nom[21];
       ...
       } **liste, elt;

char *nom_elt( void *elt);
char nouveau;
char *nom;

pkl = pkl_cre( " PICKLIST ", (void **)liste, ... );

strcpy( elt.nom, nom);

pkl_rch( pkl, elt, &nouveau, 1);

if (nouveau) liste = (struct truc **) pkl->Liste;
```

Nom: pkl_tit - Change le titre d'une "picklist"

Usage: void pkl_tit(ST_PKL *pkl, char *titre);

Prototype dans: grpkl.h

Valeur retournée: aucune

Description:
- 'pkl' : adresse de la structure ST_PKL à modifier
- 'titre' : nouveau titre,
  ou NULL pour supprimer l'ancien titre

Nom: pkl_zon - Mémorisation de la zone de l'écran ou s'ouvre la fenêtre d'une "picklist"

Usage: void pkl_zon(ST_PKL *pkl);

Prototype dans: grpkl.h

Valeur retournée: aucune

Description: Mémorise la zone de l'écran correspondant à la fenêtre de la "picklist" dans 'pkl->Memzon'

- 'pkl' : adresse de la structurte ST_PKL

S'utilise généralement juste avant pkl_fct(), et permet de retrouver l'écran initial après pkl_eff()

Poussin Jean-Christophe (1988)

Environnement d'interface utilisateur pour la conception de logiciels d'aide à la décision en agriculture

Paris : ORSTOM, 210 p. multigr.