
SOURCES DE LA FONCTION GROUPE FG10

AVERTISSEMENT

L'objet de la diffusion de ces sources est de permettre une intégration plus rapide pour les établissements et pour les sociétés de services, d'une fonction groupage identique à celle utilisée dans GENRSA et dans AGRAF.

Ainsi la documentation de ces sources n'a pas vocation à faciliter les modifications du moteur de groupage, et elle a été délibérément rédigée sans souci d'explicitations exhaustives : il s'agit d'une documentation au niveau fonctionnel uniquement. Pour chacune des fonctions utilisées, on trouvera une description de ce qu'elle exécute, sans entrer dans les détails de sa réalisation.

La description de ces fonctions ainsi que des variables globales se trouve dans les fichiers de *header*. Il s'agit de sources écrites en langage C ANSI, compilables dans les environnements UNIX, DOS et Windows.

L'intégration de ces sources dans un logiciel est soumise à la signature préalable d'un contrat d'utilisation. La conformité de leur mise en œuvre peut-être validée par la comparaison du groupage obtenu avec celui de GENRSA ou de AGRAF.

L'ATIH ne saurait être tenu responsable de modifications effectuées par d'autres qu'elle sur ces sources.

PRÉSENTATION GÉNÉRALE

La fonction groupage est monomodulaire, c'est-à-dire qu'elle s'utilise en n'appelant qu'une seule fonction, la fonction *grp_10()*. Celle-ci se charge de toutes les opérations nécessaires au groupage d'un RSS. Ces opérations sont : la lecture des RUM, le contrôle des RUM, la détermination du séjour principal, et le groupage proprement dit.

Cette documentation présente la manière d'utiliser la fonction *grp_10()*.

Type de données et structures

La fonction groupage 10 contient 4 paramètres dont seul le premier est à renseigner par l'utilisateur, les 3 autres étant calculés et remplis par la fonction groupage elle-même. Tous ces paramètres ont pour type des structures décrites ci-après :

1. Le type GRP_ENTREE_10 :

Cette structure est utilisée pour rassembler les paramètres d'entrée de la fonction groupage. Elle seul contient l'ensemble des paramètres d'entrée à passer à la fonction groupage.

```
typedef struct {
    char *rum2[tab_max];
    char dirtab[BUFSIZ];
    int ns;
    int type_etab;
} GRP_ENTREE_10;
```

***rum2[tab_max]** Tableau de pointeurs sur des chaînes de caractères. Ces chaînes contiennent les différents RUM d'un RSS. Chaque chaîne (correspondant à un RUM) doit se terminer par un caractère nul ('\0'). Ce tableau comporte au plus **tab_max** pointeurs, soit 25 éléments. Les formats acceptés sont les suivants : 011.

dirtab[BUFSIZ] Chaîne de caractères contenant le chemin d'accès absolu, complet, pour atteindre le répertoire contenant les tables. Il doit être écrit de telle sorte que l'accès à une table donnée consiste à concaténer cette chaîne avec le nom de la table recherchée.

int ns Entier contenant le nombre de séjours élémentaires du malade au cours de son séjour dans l'établissement hospitalier. Un appel à la fonction *grp_10()* avec ce paramètre à 0 entraîne la fermeture des tables de groupage.

int type_etab Entier indiquant si l'établissement est sous DGF (=1), ou sous OQN (=2).

2. Le type GRP_RESULTAT_10 :

Cette structure est calculée et renvoyée par la fonction groupage afin de fournir les résultats traditionnels du groupage comme le GHM.

```
typedef struct {
    int ns_sor;
    int sejp;
    char version[3];
    char cmd[3];
    char ghm[5];
    char cret[4];
    int ErrCtrl[MAXERRCTRL];
    int ErrArb[MAXERRARB];
    int ErrImpl[MAXERRIMPL];
} GRP_RESULTAT_10 ;
```

`int ns_sor` Entier contenant le nombre de séjours après le traitement dans la fonction groupage. Ce nombre est désormais égal au nombre de séjours en entrée.

`int sejp` entier, destinée à recevoir le numéro du RUM désigné par le groupage comme contenant le diagnostic principal et le diagnostic relié, en débutant la numérotation des RUM à 0.

`char version[3]` Chaîne de 3 caractères destinée à recevoir le numéro de version de la classification lu dans les tables.

`char cmd[3]` Chaîne de 3 caractères de long, destinée à recevoir le numéro de la CMD déterminée par le groupage (terminée par le caractère NULL).

`char ghm[5]` Chaîne de 5 caractères de long, destinée à recevoir le numéro du GHM déterminé par le groupage (terminée par le caractère NULL).

`char cret[4]` Chaîne de 4 caractères de long, destinée à recevoir le code retour déterminé par le groupage. Il s'agit de la valeur en caractère du retour de la fonction *grp_10()* (cf. plus bas). Cette chaîne est terminée par le caractère NULL.

`int ErrCtrl[MAXERRCTRL]` Tableau de **1+MAXERRCTRL** (101) entiers courts correspondant au vecteur des erreurs contenues dans chacun des RUM du RSS. Il est construit de la façon suivante :

entier n°1	: (nr) nombre de RUM détectés dans le RSS
entier n°2	: (n1) nombre d'erreurs détectées dans le RUM n°1
...	
entier n°(nr+1)	: nombre d'erreurs détectées dans le RUM n°nr
entiers n°(nr+2) à (nr+1+n1)	: codes des erreurs du RUM n°1
entiers n°(nr+n1+2) à (nr+1+n1+n2)	: codes des erreurs du RUM n°2
...	

`int ErrArb[MAXERRARB]` Tableau de **1 + MAXERRARB** (6) entiers courts correspondant au vecteur des erreurs détectées par l'arbre de décision de la classification. Les erreurs sont stockées dans ce vecteur de manière séquentielle. Ce vecteur n'a pas de structure particulière.

`int ErrImpl[MAXERRIMPL]` Tableau de **1 + MAXERRIMPL** (21) entiers courts correspondant au vecteur des erreurs dues à l'implémentation. Les erreurs sont stockées dans ce vecteur de manière séquentielle. Ce vecteur n'a pas de structure particulière.

3. Le type **GHS_VALORISATION_10**

Cette structure est calculée et renvoyée par la fonction `groupage` afin de fournir les éléments de valorisation du séjour :

```
typedef struct {
    char ghs[5];
    int nbjxsup;
    int sejxinf;

    int nbjrea;
    int nbjsrc;
    int nbjstf_issuderea;
    int nbjstf_tot;
    int nbjsra;
    int nbjssc;

    int nbac687;
    int nbac688;
    int nbac689;
    int acprlvorg;

    int pih_a;
    int pih_b;

    int nbjnna;
    int nbjnnb;
    int nbjnnc;

    char prestdialavsros[5];

    int nbhd;
    int nbent1;
    int nbent2;
    int nbent3;
    int nbchhypb;
} GVAL GHS_VALORISATION_10;
```

`char ghs[5]` Chaîne de 5 caractères de long, destinée à recevoir le numéro du GHS déterminé à partir du GHM obtenu par le groupage, et des informations contenues dans le fichier `ghsinfo.tab`. Dans le cas de la dialyse pour les établissements ex-OQN, cette variable contient le libellé du forfait dialyse après SROS (D09,D10,D11,D14,D15 ou D16). Pour les établissements ex-DGF dans le cas de la dialyse, elle contient le numéro de GHS après SROS (9500,9501,9502,9520,9521 ou 9522).

`int nbjxsup` Entier contenant le nombre de journées au-delà de la borne extrême haute pour le GHS considéré.

`int sejxinf` Entier utilisé comme booléen, contenant 0 si la durée du séjour n'est pas inférieure à la borne extrême basse du GHS ou 1 sinon.

<code>int nbjrea</code>	Entier contenant le nombre de journées de réanimation après le SROS, en suivant la définition donnée plus haut.
<code>int nbjsrc</code>	Entier contenant le nombre de journées de surveillance continue après le SROS, en suivant la définition donnée plus haut.
<code>int nbjstf_issuderea</code>	Entier contenant le nombre de journées de soins intensifs provenant de la réanimation, en suivant la définition donnée plus haut.
<code>int nbjstf_tot</code>	Entier contenant le nombre de journées de soins intensifs total, y compris les journées de soins intensifs provenant de la réanimation, en suivant la définition donnée plus haut.
<code>int nbjsra</code>	Entier contenant le nombre de journées de réanimation avant le SROS, en suivant la définition donnée plus haut.
<code>int nbjssc</code>	Entier contenant le nombre de journées de surveillance continue avant le SROS, en suivant la définition donnée plus haut.
<code>int nbac687</code>	Entier contenant le nombre d'actes supplémentaires (hors séances) de radiothérapie menant dans le GHM 28Z11Z.
<code>int nbac688</code>	Entier contenant le nombre d'actes supplémentaires (hors séances) de radiothérapie menant dans le GHM 28Z12Z.
<code>int nbac689</code>	Entier contenant le nombre d'actes supplémentaires (hors séances) de radiothérapie menant dans le GHM 28Z13Z.
<code>int acprlvorg</code>	Entier contenant la catégorie de prélèvement d'organes du séjour considéré. (0, 1, 2 ou 3).
<code>int pih_a</code>	Entier utilisé comme booléen, contenant 0 en cas d'absence de code diagnostic associé Z75.80, 1 sinon.
<code>int pih_b</code>	Entier utilisé comme booléen, contenant 0 si le mode d'entrée et le mode de sortie est égal à 0, 1 sinon.
<code>int nbjnna</code>	Entier contenant le nombre de journées de néonatalogie de type 2A.
<code>int nbjnna</code>	Entier contenant le nombre de journées de néonatalogie de type 2B, y compris les journées de type 3 sans actes de réanimation.
<code>int nbjnnc</code>	Entier contenant le nombre de journées de néonatalogie de type 3, ajouté du nombre de journées de réanimation pédiatrique.
<code>int prestdialavsros</code>	Entier contenant : pour les établissements ex-DGF, contient le numéro du GHS avant SROS (9500,9501,9502,9503) associé au GHM de dialyse obtenu . Pour les établissements ex-OQN, contient le libellé du forfait dialyse avant SROS (D01,D03,D04,D05) associé au GHM obtenu.
<code>int nbhd</code>	Entier contenant le nombre d'actes supplémentaires hors séances pour l'hémodialyse.

int ent1	Entier contenant le nombre d'actes supplémentaires hors séances pour les entraînements à la dialyse péritonéale automatisée.
int ent2	Entier contenant le nombre d'actes supplémentaires hors séances pour les entraînements à la dialyse péritonéale continue ambulatoire.
int ent3	Entier contenant le nombre d'actes supplémentaires hors séances pour les entraînements à l'hémodialyse.
int nbchhypb	Entier contenant le nombre de caissons hyperbare.

Pour les modalités de calcul de ces variables reportez-vous au chapitre :
Calcul des éléments permettant la tarification.

4. Le type **RSS_INFO_10**

Cette structure est calculée et renvoyée par la fonction `groupage` afin de fournir des éléments d'informations concernant le RSS, et également des informations propres à chaque RUM du RSS.

```
typedef struct {
    int nbdiag;
    char *listdiag;
} DIAGLIST_10;
typedef struct {
    int nbac;
    char *listzac;
} ACTLIST_10;

typedef struct {
    int brea;
    char typum[2];
    int dsp;
} INFO_RUM_10;

typedef struct {
    int agea;
    int agej;
    int dstot;
    DIAGLIST_10 diaglist ;
    ACTLIST_10 actlist;
    INFO_RUM_10 tabval[tab_max];
}RSSI RSS_INFO_10;
```

int agea	Entier contenant l'âge en années utilisé par la fonction <code>groupage</code>
int agej	Entier contenant l'âge en jours utilisé par la fonction <code>groupage</code>
int dstot	Entier contenant la durée totale de séjour calculée par la fonction <code>groupage</code>
DIAGLIST_10 diaglist	Est une structure contenant : - le nombre de diagnostics différents..

- la liste de ces diagnostics avec en première position le diagnostic principal, puis en deuxième position le diagnostic relié, puis l'ensemble des diagnostics associés retenus pour le RSS.

ACTLIST_10 actlist Est une structure contenant :

- le nombre de zones d'actes du RSS.
- la liste des zones d'actes .

La présentation de ces zones d'actes est identique à celle du RUM

INFO_RUM_10 tabval[tab_max] est un tableau de tab_max éléments contenant pour chaque RUM le type d'unité médicale (variable typum), la durée de séjour partielle (variable dsp) ainsi qu'un booléen indiquant la présence de journée de réanimation (Actes marqueurs+IGS >= borne, variable brea).

Syntaxe de l'appel de la fonction grp_10()

Seul le paramètre grp_ent doit être renseigné avec l'appel :

```
cr=grp_10(grp_ent_10,&grp_res_10,&ghs_val_10,&rss_inf_10)
```

où :

```
GRP_ENTREE_10 grp_ent_10;
GRP_RESULTAT_10 grp_res_10;
GHS_VALORISATION_10 ghs_val_10;
RSS_INFO_10 rss_inf_10;
```

Valeur en retour (code-retour)

La fonction *grp_10()* renvoie un entier égal à zéro si les contrôles n'ont rien détecté d'anormal, et si le groupage n'a pas trouvé d'anomalies. Dans le cas contraire, la valeur retournée est le numéro de l'erreur détectée (sauf pour certaines erreurs de contrôles non bloquantes non retournées par la fonction groupage).

Ce code-retour est également placé, sous forme d'une chaîne, dans la structure GRP_RESULTAT_10 passée en deuxième paramètre à la fonction *grp_10()*.

Toute erreur de contrôle dont la valeur est contenue dans la liste erreursbloq_10 du fichier fg10.h est irrémédiable, et ne permet pas d'obtenir le groupage du R.S.S. Les erreurs de contrôle non contenues dans cette liste ne sont pas graves en terme de groupage (par exemple : date système improbable). De telles erreurs ne sont détectées que par l'analyse du vecteur décrit plus haut (ErrCtrl).

Les erreurs d'implémentation (vecteur ErrImpl) sont aussi irrémédiables, ainsi que les erreurs de groupage (tableau ErrArb), excepté pour l'erreur de groupage 80.

Compte tenu de la numérotation particulière des erreurs (trois catégories) l'exploitation du code retour seul nécessite de connaître le résultat du « groupage » : 90Z03Z pour une erreur d'implémentation bloquante, 90Z00Z pour un contrôle bloquant, 90C01Z, 90Z01Z ou 90Z02Z pour une erreur bloquante dans l'arbre décisionnel (erreur de groupage), tout autre groupe ou GHM pour un résultat correct (et dans ce cas, code-retour nul ou ayant une valeur non contenue dans la liste erreursbloq_10).

