COMPTE RENDU DU TP1 DE SDD

Binôme :

AZEROUAL Mohammed

LINGOM NKAMGA David Cédric

Sommaire

1 Présentation générale 3
1.1 Description de l'objet du TP 3
1.2 Description de la structure de données 3
1.3 Schéma de la liste chainées à deux niveaux 4
1.4 Schéma de la liste contigüe de jours 5
1.5 Description des fichiers des données utilisées (en entrée et en sortie) 6
1.6 Organisation du code source 6
2 Détail de chaque fonction 8
3 Compte rendu d'exécution 49
3.1 Makefile 49
3.2 Jeux de test complets 50
3.2.1 Création de la liste chainée triée 50
3.2.2 Création d'une liste contigüe de jours 56
3.2.3 Suppression d'une action dans la liste chainée à deux niveaux 59

1 Présentation générale

1.1 Description de l'objet du TP

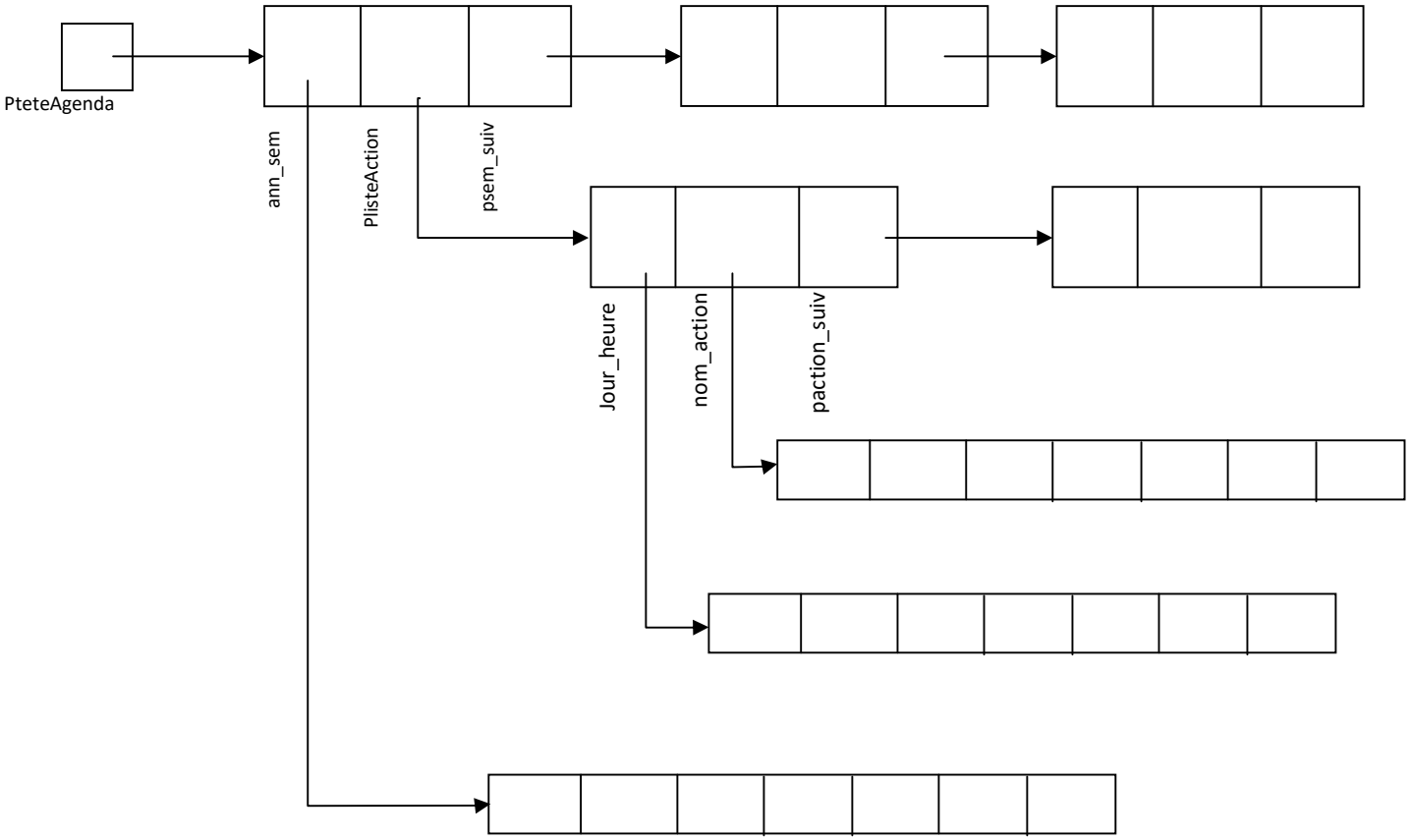
Dans ce TP nous sommes emmener à travail sur la structure de donnée qui est la liste chainée mais plus précisément dans notre cas nous allons devoir créer une liste chainées à deux niveaux et triée de données qui sont lu à partir d'un fichier texte. Après la création de cette liste chainée à deux niveaux, les données de cette liste son stocker dans un autre fichier texte de sauvegarde. Plus tard cette liste de deux niveaux nous permettra de créer une liste contigüe de jours associer à des actions qui contiennent un motif rechercher. Mais aussi, hors mi cela nous avons grâce cette liste chainée à deux niveau nous aurons la possibilité de supprimer une action connaissant l'année, la semaine, le jour et l'heure. Sans oublier que nous sommes sensé prendre en considération le fait qu'il y à pas de semaine vide.

1.2 Description de la structure de données

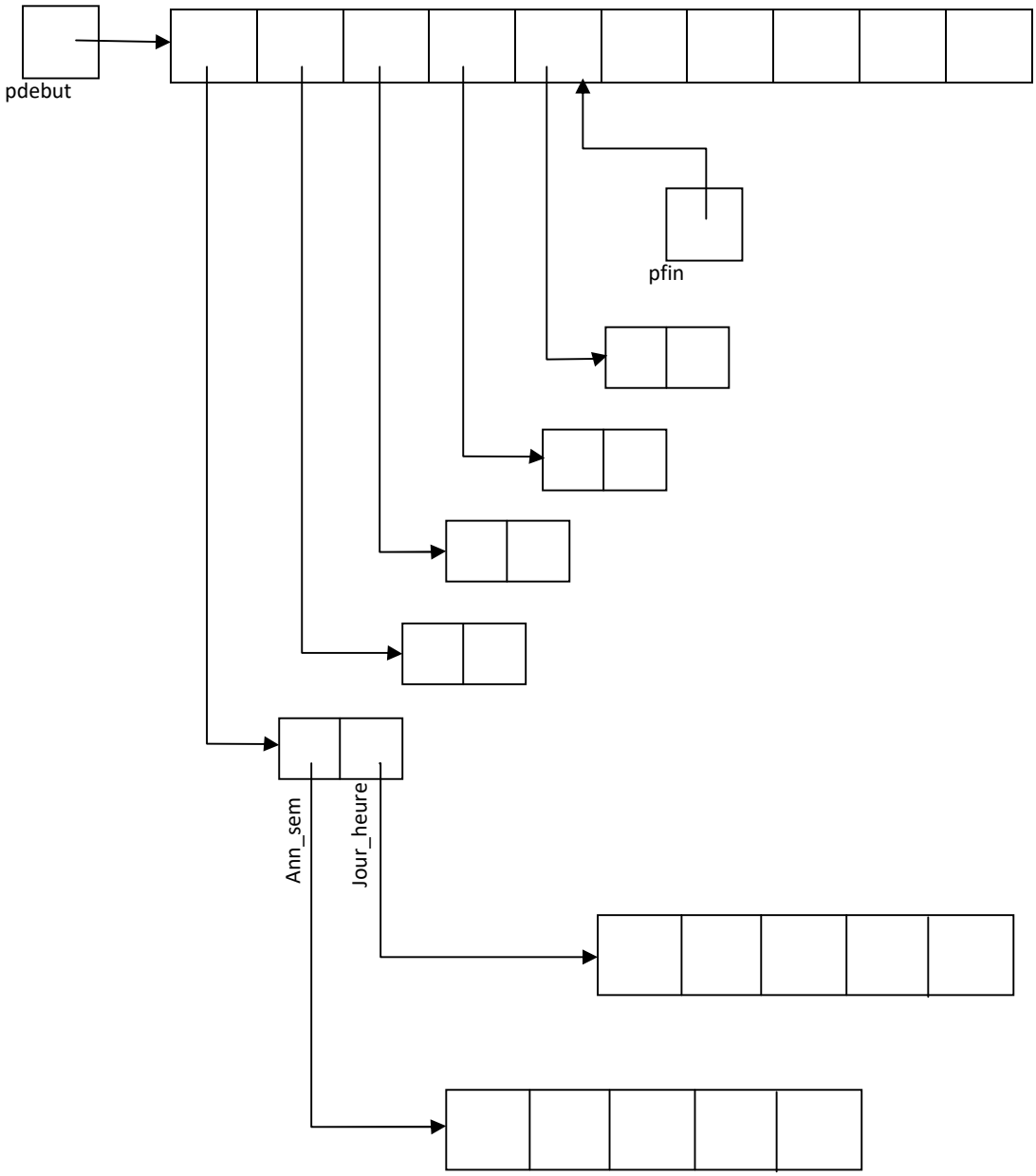
Comme déjà dis plus haut les structures de données utilisé sont : la liste contigüe et la liste chainée plus de quoi une liste chainée à deux niveaux. Le premier niveau de la liste chainée est constituées des éléments qui sont des structure à 3 champs qui sont: une chaine de caractères contenant l'année et semaine , un pointeur vers une liste actions de type action_t * et un pointeur vers l'élément suivant de la liste du premier niveau de type semaine_t *. En ce qui concernent le deuxième niveaux qui est une sous liste chainée associée à une semaine tel que chaque éléments de cette liste est une structure de type action_t constituer de trois champs. Ces trois champs sont une chaine de caractères censé contenir le jour et l'heure d'une action, une chaine de caractères contenant l'action et un pointeur sur l'action suivante de type action-t *.

Pour la liste contigüe nous utiliserons un tableau avec une TAILLE_MAX qui contiendra nos éléments contigües. Les éléments qui compose ce tableau sont des pointeur sur la structure de type jour_t composé de deux champs qui sont: un tableau de caractère qui contiendra l'année et la semaine et un autre tableau de qui contiendra le jour et l'heure. Ainsi, avec cette structure on pourra avoir des informations précise permettant de savoir l'année, la semaine, le jour et l'heure des actions contenant un motif rechercher.

1.3 Schéma de la liste chainées à deux niveaux



1.4 Schéma de la liste contigüe de jours



1.5 Description des fichiers des données utilisées (en entrée et en sortie)

La lecture des données vont être fait à partir des fichiers lecture1.txt, lecture2.txt, lecture3.txt et lecture4.txt et cela sera passé en ligne de commande. La sauvegarde quand à elle se fera dans un fichier texte appelé sauvegarde.txt

1.6 Organisation du code source

Notre code source constitué de fichier.c et fichier.h qui sont comme suit:

Nous avons le fichiers principal main.c qui contient notre fonction principale qui fait appelé à des fonctions liées au traitement de création, sauvegarde, suppression. Dans ce fichiers source nous avons comme fichier d'entête "listeContigue .h". Notre fichier listeContigue.h est aussi inclus dans le fichier listeContigue.c ce qui contient les fonctions liées à la manipulation de liste contigüe de jours dont les prototypes sont comme suit:

```
int CompareChaine(char *, char *, int);

int RechercherMotif(char *, char *);

jour_t ** AllocationListeContigue ();

jour_t * AllocationJour();

void CreerJour(jour_t *, semaine_t *, action_t *);

void InsererJour(jour_t **, jour_t *,int);

void LectureSemaineJour(jour_t *, int *);

void LectureMotif(char **, int *);

void CreerListeJour(semaine_t *, char *, jour_t **,jour_t ***);

void AfficherListeContigue(jour_t **, jour_t **);

void LibererListeContigue(jour_t ***, jour_t ***);
```

Ensuite dans le fichiers d'entête listeContigue.h est inclus le fichier d'entête listeChainee_semaine.h qui elle est aussi inclus dans le listeChainee_semaine.c où sont définies les fonctions et procédures liées à la manipulations du premier niveau de la liste chainée c'est-à-dire liste chainée des semaines, lecture et sauvegarde dans un fichier qui sont comme suit:

```
semaine_t * AllocationSemaine();

void CreerSemaine (semaine_t *, char *);

void InsertionSemaine (semaine_t **, semaine_t *);

semaine_t **  RechercherSemaine (semaine_t **, char *, int *);
```

```
void SuppressionSemaine(semaine_t **);

void LibererListe(semaine_t **);

void CreerListe (semaine_t **, FILE *);

void LectureFichier(char *, semaine_t **, int *);

void SauvegardeDansFichier(semaine_t *, int *);

void AfficheSauvegarde();

void supression(semaine_t **, char *, char *);
```

Outre cela dans le fichier d'entête listeChainee_semaine.h est inclus le fichier d'entête listeChainee_action.h qui est aussi dans la listeChainee_action.c qui défini les fonctions et procédures liées à la manipulation et au traitement de la liste chainée de deuxième niveau c'est-à-dire la liste chainée des actions par rapport à chaque semaine présente dans la liste chainée qui sont comme suit:

```
ction_t * AllocationAction ();

void CreerAction (action_t *, char *, char *);

action_t ** RechercherAction (action_t **, char *, int *);

void InsertionAction (action_t **,action_t *);

void SuppressionAction(action_t **);
```

2 Détail de chaque fonction

Au sujet des fonctions utilisés dans ce TP qui repartis dans les différents fichiers .c , nous allons vous donner les détails des fonctions de chaque fichier.c.

Les fonctions et procédures du fichiers listeChainee_ action.c sont détaillés comme suit:

```
/*-------------------------------------------------------------------------------------------------------------*/
/*                                          listeChainee_action.c                                              */
/*                                                                                                             */
/* Rôle:   Définition des fonctions qu'on utilise pour la création d'une liste chainée d'actions               */
/*         triée à partir d'un fichier de données  où son nom est donné en ligne de commande.                  */
/*-------------------------------------------------------------------------------------------------------------*/

#include "./listeChainee_action.h"

/*-------------------------------------------------------------------------------------------------------------*/
/* AllocationAction       Alloue le bloc action.                                                               */
/*                                                                                                             */
/* En entrée:   Rien en entrée                                                                                 */
/*                                                                                                             */
/* En sortie:   On retourne l'adresse du bloc action alloué.                                                   */
/*                                                                                                             */
/* variables locales:      paction - Pointeur sur le bloc alloué.                                              */
/*-------------------------------------------------------------------------------------------------------------*/
```

```
action_t * AllocationAction ()
{
	action_t * paction=(action_t *)malloc(sizeof(action_t));
	return paction;
}
```

```
/*-----------------------------------------------------------------------------------------------------------------*/
/* CreerAction	Crée l'action en introduisant dans le bloc alloué le jour et l'heure		*/
/*			lu à partir d'un fichier.					*/
/*									*/
/* En entrée:	paction - Pointeur sur le bloc alloué pour l'action.		*/
/*			s1 - Chaine de caractères contenant le jour et l'heure de l'action.	*/
/*			s2 - Chaine de caractère contenant le nom de l'action.		*/
/*									*/
/* En sortie:	Rien en sortie						*/
/*-----------------------------------------------------------------------------------------------------------------*/
```

```
void CreerAction (action_t * paction, char * s1, char * s2)
{
	strcpy(paction->jour_heure,s1);		/*copie le jour et l'heure*/
	strcpy(paction->nom_action,s2);		/*copie le nom de l'action*/
	paction->paction_suiv=NULL;
}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* RechercherAction         Recherche une action dans la liste chainée des actions.                                  */
/*                                                                                                                   */
/* En entrée:  PpteteListe -  Pointeur de pointeur de tête de la liste chainée des actions.                          */
/*                      pvaleur  -  Pointeur sur l'action à rechercher (qui est une chaine de caractères).           */
/*                   ptrouver    -  Pointeur sur une case mémoire contenant 0 si on a pas trouvé ou                  */
/*                                  si on a trouvé la chaine.                                                        */
/*                                                                                                                   */
/* En sortie:        prec - Pointeur de pointeur de tete de liste chainée des actions ou pointeur sur la*/
/*                          case pointeur d'élément précédent dans la liste chainée d'actions.                       */
/*                ptrouver - Pointeur sur une case mémoire contenant 0 si on a pas trouvé ou 1 si on a */
/*                          trouvé l'action dans la liste chainée des actions.                                       */
/*                                                                                                                   */
/* Variable(s) locale(s) :   pcour - Pointeur sur l'action courante.                                                 */
/*                             prec  - Pointeur de pointeur de tete de liste chainée des actions ou                  */
/*                                     pointeur sur la case pointeur de l'élément précédent de la                    */
/*                                     liste chainée des actions.                                                    */
/*-------------------------------------------------------------------------------------------------------------------*/
```

```
action_t ** RechercherAction (action_t ** PpteteListe, char * pvaleur, int * ptrouver)

{

	action_t * pcour = *PpteteListe, ** prec = PpteteListe;	/*Initialisation à la première action et au pointeur de tete*/

	*ptrouver=0;

	while ((pcour != NULL) && (strcmp(pvaleur,pcour->jour_heure) > 0))	/*Tantque je suis dans la liste et que ma chaine est plus grande*/

	{

		prec = &(pcour->paction_suiv);	/*on récupère l'adresse de la case pointeur de l'élément courant*/

		pcour = *prec;					/*passe au suivant*/

	}

	if ((pcour != NULL) && (!strncmp(pvaleur, pcour->jour_heure,TAILLE_JOUR_HR)))	/*si on trouve la chaine*/

	{

		*ptrouver=1;

	}

	return prec;

}
```

```
/*-------------------------------------------------------------------------------------------------------------- -*/
/* InsertionAction        Insère une nouvelle action dans la liste chainée des actions.                        */
/*                                                                                                             */
/* En entrée:    ppaction - Pointeur de pointeur de tete de liste chainée des actions ou pointeur     */
/*                                  sur la case pointeur de l'élément précédent.                               */
/*                       paction - Pointeur sur l'action à insérer.                                            */
/*                                                                                                             */
/* En sortie:    ppaction - Pointeur de pointeur de tete de liste chainée des actions ou pointeur      */
/*                                  sur la case pointeur de l'élément précédent.                               */
/*-------------------------------------------------------------------------------------------------------------*/



void InsertionAction (action_t ** ppaction,action_t * paction)
{
        paction->paction_suiv = *ppaction;
        *ppaction = paction;
}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* SuppressionAction        Supprime une action de liste chainée des actions.                                        */
/*                                                                                                                   */
/* En entrée:   ppaction - Pointeur de pointeur de tete de liste chainée des actions ou pointeur sur                 */
/*                         la case pointeur de l'élément précédent de la liste chainée des actions.                  */
/*                                                                                                                   */
/* En sortie:      ppaction - Pointeur de pointeur de tete de liste chainée des actions ou pointeur                  */
/*                            sur la case pointeur de l'élément précédent de la liste chainée des                    */
/*                            actions                                                                                */
/*                                                                                                                   */
/* Variable(s) locale(s):  paction - Pointeur sur l'action à supprimer.                                              */
/*-------------------------------------------------------------------------------------------------------------------*/


void SuppressionAction(action_t ** ppaction)
{
        action_t * paction = *ppaction;         /*recupère l'adresse de l'action à supprimer*/
        *ppaction = paction->paction_suiv;          /*l'action précédente pointe sur l'action après l'action à supprimer*/
        free(paction);                      /*supprime l'action*/
        paction = NULL;
}
```

Les fonctions et procédures du fichiers listeChainée_semaine.c sont détaillés comme suit:

```
/*-------------------------------------------------------------------------------------------------------------*/
/*                                    listeChaine_semaine.c                                                    */
/*                                                                                                             */
/* Role:  Définition des fonctions permettent de manipuler la listeChainée des semaines et les     */
/*          actions associées à une année, semaine, jour et heure particulière.                                */
/*-------------------------------------------------------------------------------------------------------------*/

#include "./listeChainee_semaine.h"



/*-------------------------------------------------------------------------------------------------------------*/
/* AllocationSemaine        Alloue un bloc semaine où sera stocker la semaine.                                 */
/*                                                                                                             */
/* En entrée:   Rien en entrée                                                                                 */
/*                                                                                                             */
/* En sortie:   On retourne l'adresse du bloc semaine alloué.                                                  */
/*                                                                                                             */
/* Variable(s) locale(s):   psemaine - Pointeur sur le bloc alloué.                                            */
/*-------------------------------------------------------------------------------------------------------------*/
```

```
semaine_t * AllocationSemaine()
{
        semaine_t * psemaine = (semaine_t *)malloc(sizeof(semaine_t));
         return psemaine;
}



/*-------------------------------------------------------------------------------------------------------------------------*/
/* CreerSemaine      Crée la semaine en introduisant dans le bloc alloué l'année et le numéro de   */
/*                    semaine lu à partir d'un fichier.                                            */
/*                                                                                                 */
/* En entrée:   psemaine - Pointeur sur le bloc alloué pour la semaine.                            */
/*                        s - Chaine de caractères contenant l'année et la semaine.                */
/*                                                                                                 */
/* En sortie:   Rien en sortie                                                                     */
/*-------------------------------------------------------------------------------------------------------------------------*/



void CreerSemaine(semaine_t * psemaine, char * s)
{
        strcpy(psemaine->ann_sem,s);
         psemaine->PlisteAction=NULL;
         psemaine->psem_suiv=NULL;
}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* InsertionSemaine      Insère une nouvelle semaine dans la liste chainée des semaines.                             */
/*                                                                                                                   */
/* En entrée:   ppsemaine - Pointeur de pointeur de tete de liste chainée des semaines ou pointeur                   */
/*                                  sur la case pointeur de l'élément précédent.                                     */
/*                      psemaine - Pointeur sur la semaine à insérer.                                                */
/*                                                                                                                   */
/* En sortie:   ppsemaine - Pointeur de pointeur de tete de liste chainée des semaines ou pointeur                   */
/*                                  sur la case pointeur de l'élément précédent.                                     */
/*-------------------------------------------------------------------------------------------------------------------*/



void InsertionSemaine (semaine_t ** ppsemaine,semaine_t * psemaine)
{
        psemaine->psem_suiv = *ppsemaine;
        *ppsemaine = psemaine;
}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* SupressionSemaine         Supprime une semaine de liste chainée des semaines.                                     */
/*                                                                                                                   */
/* En entrée:    ppsemaine -  Pointeur de pointeur de tete de liste chainée des semaines ou                          */
/*                            pointeur case pointeur de l'élément précédent de la liste chainée des                  */
/*                            semaines.                                                                              */
/*                                                                                                                   */
/* En sortie:    ppsemaine -  Pointeur de pointeur de tete de liste chainée des semaines ou                          */
/*                            pointeur sur la case pointeur de l'élément précédent de la liste                       */
/*                            chainée des semaines.                                                                  */
/*                                                                                                                   */
/* Variable(s) locale(s):    psemaine - Pointeur sur la semaine à supprimer.                                         */
/*-------------------------------------------------------------------------------------------------------------------*/



void SuppressionSemaine(semaine_t ** ppsemaine)
{
        semaine_t * psemaine = *ppsemaine;                  /*recupère l'adresse de la semaine à supprimer*/
        *ppsemaine = psemaine->psem_suiv;                   /*la semaine précédente pointe sur la semaine après la semaine à supprimer*/
         free(psemaine);                                    /*supprime la semaine*/
         psemaine = NULL;
}
```

```
/*-------------------------------------------------------------------------------------------------------------------------*/
/* LibererListe            Libère toute la liste à deux niveau.                                                            */
/*                                                                                                                         */
/* En entrée:  ppsemaine - Pointeur de pointeur de tete de liste chainée.                                                  */
/*                                                                                                                         */
/* En sortie:  ppsemaine - Pointeur de pointeur de tete de liste chainée.                                                  */
/*-------------------------------------------------------------------------------------------------------------------------*/



void LibererListe(semaine_t ** ppsemaine)
{
        while ((*ppsemaine) != NULL)                    /*libère la liste chainée des semaines*/
        {
                while((*ppsemaine)->PlisteAction != NULL)
                {
                        SuppressionAction(&((*ppsemaine)->PlisteAction));   /*libère la liste chainée des actions associés à chaque semaine*/
                }
                SuppressionSemaine(ppsemaine);
        }
}
```

```
/*-------------------------------------------------------------------------------------------------------------------------*/
/* RechercherSemaine       Recherche une semaine dans la liste chainée des semaines.                                       */
/*                                                                                                                         */
/* En entrée:     PpteteListe - Pointeur de pointeur de tête de la liste chainée des semaines.                             */
/*                      pvaleur - Pointeur sur la semaine à rechercher                                                     */
/*                                (qui est une chaine decaractères)                                                        */
/*                      ptrouver - Pointeur sur une case mémoire contenant 0 si on a pas trouvé ou 1                       */
/*                                 si on a trouvé la chaine                                                                */
/*                                                                                                                         */
/* En sortie:          prec - Retourne l'adresse du pointeur de tete de liste chainée des semaines ou                      */
/*                            l'adresse de la case pointeur de l'élément précédent dans la liste chainée                   */
/*                            de semaines.                                                                                 */
/*                ptrouver - Pointeur sur une case mémoire contenant 0 si on a pas trouvé l'élément                        /
/*                           ou 1 si on a trouvé dans la liste chainée des semaines.                                       */
/*                                                                                                                         */
/* Variable(s) locale(s):   pcour - Pointeur sur la semaine courante.                                                      */
/*                            prec  - Pointeur de pointeur de tete de liste chainée des semaines ou                        */
/*                                    pointeur sur la case pointeur de l'élément précédent de la liste                     */
/*                                    chainée des semaines.                                                                */
/*-------------------------------------------------------------------------------------------------------------------------*/
```

```
semaine_t **  RechercherSemaine (semaine_t ** PpteteListe,char * pvaleur, int * ptrouver)

{

        semaine_t * pcour = *PpteteListe, ** prec = PpteteListe;          /*Initialisation à la première semaine et au pointeur de tete*/

         *ptrouver = 0;

         while   ((pcour != NULL) && (strcmp(pvaleur, pcour->ann_sem) > 0)) /*Tantque je suis dans la liste et que ma chaine est plus grande*/

        {

        prec = &(pcour->psem_suiv);                         /*On récupère l'adresse de la case pointeur de l'élément courant*/

         pcour = *prec;                                  /*Passe au suivant*/

        }

         if ((pcour != NULL) && (!strcmp(pvaleur, pcour->ann_sem)))       /*Si on trouve la chaine*/

        {

                *ptrouver=1;

        }

        return prec;

}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* CreerListe       Cree la liste chainée à deux niveaux triée à partir des lignes lu dans fichier.          */
/*                                                                                                           */
/* En entrée:   PpteteListe - Pointeur de pointeur de la tete de liste à deux niveaux.                       */
/*                         f - Fichier à partir du quel on va lire.                                          */
/*                                                                                                           */
/* En sortie:   PpteteListe - Pointeur de pointeur de la tete de liste à deux niveaux.                       */
/*                                                                                                           */
/* Variable(s) locale(s):       trouver - Permet de dire si on a trouver l'élément qu'on recherche ou        */
/*                                 pas.                                                                      */
/*                                      s1 - Tableau de caractère qui va contenir l'année et semaine         */
/*                                           lu à partir du fichier.                                         */
/*                                      s2 - Tableau de caractère qui va contenir le jour et l'heure         */
/*                                           lu à partir du fichier.                                         */
/*                                       s3 - Tableau de caractères qui va contenir l'action lu à            */
/*                                            partir du fichier.                                             */
/*                                  psemaine - Pointeur sur la semaine à insérer.                            */
/*                               PrecSemaine - Pointeur de pointeur de tete de liste des semaines ou         */
/*                                           pointeur sur la case pointeur de l'élément précédent de la      */
/*                                           liste des semaines.                                             */
/*                                  paction - Pointeur sur l'action à insérer.                               */
/*                               PrecAction - Pointeur de pointeur de tete de liste des actions ou           */
/*                                            pointeur sur la case pointeur de l'élément précédent de la     */
/*                                           liste des actions.                                              */
/*                                                                                                           */
/*---------------------- -------------------------------------------------------------------------------------------------------------*/
```

```
void CreerListe (semaine_t ** PpteteListe, FILE * f)
{
	int trouver;
	char s1[TAILLE_SEMAINE];
	char s2[TAILLE_JOUR_HR];
	char s3[TAILLE_ACTION];
	semaine_t * psemaine = NULL, ** PrecSemaine = NULL;
	action_t * paction = NULL, ** PrecAction = NULL;
	while (fgets(s1,TAILLE_SEMAINE,f) != NULL)		/*Test si je ne suis pas à la fin du fichier*/
	{
		trouver=0;
		if (fgets(s2,TAILLE_JOUR_HR,f) != NULL)
		{
			PrecSemaine = RechercherSemaine (PpteteListe,s1, &trouver);
			 if (!(trouver))			/*Si j'ai pas trouver la semaine rechercher*/
			{
				psemaine = AllocationSemaine();
				if (psemaine != NULL)	/* Vérifie si l'allocation semaine c'est bien passé*/
				{
					CreerSemaine (psemaine,s1);
					InsertionSemaine(PrecSemaine,psemaine);
				}
			}
			if (fgets(s3,TAILLE_ACTION,f) != NULL)
			{
				trouver = 0;
```

```
                                PrecAction = RechercherAction (&((*PrecSemaine)->PlisteAction), s2,
&trouver);
                                if (!(trouver))                                 /*Si j'ai trouver l'action
rechercher*/
                                {
                                        paction = AllocationAction();
                                        if (paction != NULL)                            /*Vérifier si
l'allocation de l'action c'est bien passé*/
                                        {
                                                CreerAction (paction, s2, s3);
                                                InsertionAction(PrecAction, paction);
                                        }
                                        else
                                        {
                                                LibererListe(PpteteListe);
                                        }
                                }
                        }
                }
        }
}
```

```
/*-------------------------------------------------------------------------------------------------------------------------*/
/* LectureFichier                Lit chaque ligne de notre fichier qui contient comme données année,*/
/*                               semaine, jour,  heure et créer une liste chainée à deux niveaux à       */
/*                               partir de cette lecture.                                                */
/*                                                                                                       */
/* En entrée:     NomFichier - Nom du fichier dans le quel on va lire.                                   */
/*                PpteteListe - Pointeur de pointeur de tete de liste chainée des semaines.              */
/*              PcodeLecture - Pointeur sur la case contenant le code de lecture c'est-à-dire 1 si la    */
/*                               lecture c'est  bien passé et 0 sinon.                                   */
/*                                                                                                       */
/* En sortie:    PpteteListe - Pointeur de pointeur de tete de liste chainée des semaines.               */
/*          PcodeLecture - Pointeur sur la case contenant le code de lecture c'est-à-dire 1 si la        */
/*                               lecture c'est bien passé et 0 sinon.                                    */
/*                                                                                                       */
/* Variable(s) locale(s):  f - Pointeur sur le fichier.                                                  */
/*-------------------------------------------------------------------------------------------------------------------------*/
```

```
void LectureFichier(char * NomFichier, semaine_t ** PpteteListe, int * PcodeLecture)
{
	FILE * f = NULL;
	f = fopen(NomFichier,"r");
	if (f != NULL)
	{
		CreerListe(PpteteListe, f);			/*Créer la liste à deux niveaux.*/
		fclose(f);
	}
	else					/*Si la lecture ne s'est pas bien passé*/
	{
		*PcodeLecture = 0;
	}
}
```

```
/*-------------------------------------------------------------------------------------------------------------------------*/
/* SauvergardeDansFichier	Sauvegarde de la liste chainée à deux niveaux triée dans un fichier. */
/*                                                                                                                         */
/* En entree:	PteteListe - Pointeur de tete de liste chainées des semaines.                                              */
/*		PcodeSauvegarde - Pointeur sur la case contenant le code de sauvegarde qui vaut 1                  */
/*				c'est-à-dire la sauvgarde est supposer vrai en entrée                                      */
/*                                                                                                                         */
/* En sortie:	PcodeSauvegarde - Pointeur sur la case contenant le code de sauvegarde                             */
/*				c'est-à-dire 1 si la sauvgarde c'est bien passer et 0 sinon.                               */
/*                                                                                                                         */
```

```
/* Variable(s) locale(s):   pcour1 - Pointeur sur la semaine courante.                                   */
/*                          pcour2 - Pointeur sur l'action courante.                                    */
/*                                f - Pointeur sur le fichier de sauvegarde.                            */
/*--------------------------------------------------------------------------------------------------------*/



void SauvegardeDansFichier(semaine_t * PteteListe, int * PcodeSauvegarde)
{
        semaine_t * pcour1 = PteteListe;
        action_t *pcour2;
        FILE * f = fopen("sauvegarde.txt","w");
        if(f != NULL)
        {
                while(pcour1 != NULL)                                   /*tantque je suis dans la
liste des semaines*/
                {
                        pcour2 = pcour1->PlisteAction;
                        while   (pcour2 != NULL)                        /*tantque je suis
dans la liste d'actions*/
                        {
                                fprintf(f,"%s%s%s",  pcour1->ann_sem,  pcour2->jour_heure,  pcour2-
>nom_action);
                                pcour2 = pcour2->paction_suiv;          /*passe à
l'action suivante*/
                        }
                        pcour1 = pcour1->psem_suiv;                     /*passe à la
semaine suivante*/
                }
                fclose(f);
```

```
        }
        else
        {
                *PcodeSauvegarde = 0;
        }
}


/*-------------------------------------------------------------------------------------------------------------------------------/
/* AfficheSauvegarde   Affiche les année, semaine, jour, heure et action qui ont été sauvegarder.     */
/*                                                                                                     */
/* En entrée:    Rien en entrée.                                                                       */
/*                                                                                                     */
/* En sortie:    Rien en sortie.                                                                       */
/*                                                                                                     */
/* Variable(s) locale(s):    TailleChaine - Taille de chaine de carctères qui représente celle de la   */
/*                                          ligne lu dans le fichier.                                  */
/*                                  chaine - Tableau de caractères.                                    */
/*                                      f - Pointeur sur le fichier de sauvegarde.                     */
/*-------------------------------------------------------------------------------------------------------------------------------*/
```

```
void AfficheSauvegarde()

{

	int TailleChaine = TAILLE_SEMAINE+TAILLE_JOUR_HR+TAILLE_ACTION-3;

	char chaine[TAILLE_SEMAINE+TAILLE_JOUR_HR+TAILLE_ACTION-3];	/*permet de récupérer chaque ligne du fichier*/

	FILE * f = fopen("./sauvegarde.txt","r");

	if(f != NULL)

	{

		while (fgets(chaine,TailleChaine, f) != NULL)		/*tantque je ne suis pas à la fin du fichier*/

		{

			printf("%s", chaine);

		}

		fclose(f);

	}

}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* supression        Supprime une action connaissant l'année, la semaine, le jour et l'heure.                       */
/*                                                                                                                   */
/* En entrée:    PpteteListe - Pointeur de pointeur de tete de la liste chainée à deux niveaux.                      */
/*                       ann_sem - Pointeur sur la chaine de caractères qui contiendra l'année et la                 */
/*                                 semaine                                                                           */
/*                     jour_heure - Pointeur sur la chaine de caractères qui contiendra le jour et l'heure.          */
/*                                                                                                                   */
/* En sortie:    PpteteListe - Pointeur de pointeur de tete de la liste chainée à deux niveaux.                      */
/*                                                                                                                  */
/* Variable(s) locale(s):   PrecSemaine - Pointeur de pointeur de tete de liste des semaines ou                     */
/*                                        pointeur sur la case pointeur de l'élément précédent de la                 */
/*                                        liste des semaines.                                                        */
/*                                PrecAction - Pointeur de pointeur de tete de liste des semaines ou                 */
/*                                        pointeur sur la case pointeur de l'élément précédent de la                 */
/*                                        liste des semaines.                                                        */
/*-------------------------------------------------------------------------------------------------------------------*/
```

```
void supression(semaine_t ** PpteteListe, char * ann_sem, char * jour_heure)
{
	int trouver, CodeSauvegarde = 0;
	semaine_t ** PrecSemaine = NULL;
	action_t ** PrecAction =NULL;
	if(*PpteteListe != NULL)			/*si liste chainée des semaines non vide*/
	{
		PrecSemaine = RechercherSemaine(PpteteListe, ann_sem, &trouver);
		if (trouver)				/* si j'ai trouver la semaine rechercher*/
		{
			PrecAction = RechercherAction(&((*PrecSemaine)->PlisteAction), jour_heure, &trouver);
			if(trouver)				/* si j'ai trouver l'action rechercher*/
			{
				SuppressionAction(PrecAction);
				if((*PrecSemaine)->PlisteAction == NULL)
				{
					SuppressionSemaine(PrecSemaine);
				}
			}
		}
		SauvegardeDansFichier(*PpteteListe, &CodeSauvegarde);	/*sauvegarde après suppression*/
		printf("Le contenu de la sauvegarde après supression\n");
		AfficheSauvegarde();			/*Affiche la nouvelle sauvegarde*/
	}
}
```

Les fonctions et procédures du fichiers listeContigue.c sont détaillés comme suit:

```
/*-------------------------------------------------------------------------------------------------------------------*/
/*                                              listeContigue.c                                                      */
/*                                                                                                                   */
/* Role : Définition des fonctions permettant de créer la liste contigue des jours à partir d'un motif */
/*          contenu dans une action et la manipulation cette liste contigue.                                         */
/*-------------------------------------------------------------------------------------------------------------------*/


#include "./listeContigue.h"


/*-------------------------------------------------------------------------------------------------------------------*/
/* CompareChaineN    Compare deux chaine de caractères.                                                              */
/*                                                                                                                   */
/* En entrée:    s1, s2 - Deux chaines de caractères.                                                                */
/*                  taille - la taille des deux chaine de caractères.                                                */
/*                                                                                                                   */
/* En sortie:    Retourne une valeur entière taille si les deux chaines de caractères sont égaux et                  */
/*                  une valeur strictement inférieur à taille s'ils sont différent.                                  */
/*                                                                                                                   */
/* Variable(s) locale(s):    i - variable de boucle.                                                                 */
/*-------------------------------------------------------------------------------------------------------------------*/
```

```
int CompareChaine(char * s1, char * s2, int taille)
{
        int i=0;
        while ((i<taille) && (s1[i] == s2[i]))
        {
                ++i;
        }
        return i;
}

/*-------------------------------------------------------------------------------------------------------------------*/
/* RechercherMotif    Recherche un motif dans une action.                                                            */
/*                                                                                                                   */
/* En entrée:   NomAction - Chaine de caractères contenant le nom de l'action.                                       */
/*                          motif - Chaine de caractères contenant le motif.                                         */
/*                                                                                                                   */
/* En sortie:   Retourne la taille du motif si sa trouve le motif et un une valeur strictement inférieur            */
/*              à la taille du motif si on trouve pas le motif.                                                      */
/*                                                                                                                   */
/* Variable(s) locale(s):        TailleMotif - La taille du motif.                                                   */
/*                               NbCompare - Le nombre de caractères du motif comparé.                               */
/*                                   decalage - Décalage paraport au premier caractère de l'action.                  */
/*                           TailleResteAction - Le reste de la chaine action dans le quel on doit                   */
/*                                               rechercher le motif.                                                */
/*-------------------------------------------------------------------------------------------------------------------*/
```

```
int RechercherMotif(char * NomAction, char * motif)
{
	int TailleMotif = strlen(motif),NbCompare = 0, decalage=0, TailleResteAction = (TAILLE_ACTION-2);
	while ((TailleMotif <= TailleResteAction) && (NbCompare < TailleMotif))		/* tantque je peux encore rechercher le motif*/
	{
		NbCompare = CompareChaine(NomAction+decalage, motif, TailleMotif);
		if (!NbCompare)		/*si le premier caractère du motif n'est pas trouver*/
		{
			TailleResteAction -= NbCompare+1;
			decalage += NbCompare+1;
		}
		else
		{
			TailleResteAction -= NbCompare;
			decalage += NbCompare;
		}
	}
	return NbCompare;
}
```

```
/*-------------------------------------------------  ---------------------------------------------------------------------*/
/* AllocationListeContigue  Allocation d'un tableau de pointeur de jour.                                                  */
/*                                                                                                                        */
/* En entrée:    Rien en entrée.                                                                                          */
/*                                                                                                                        */
/* En sortie:    Retourne l'adresse du tableau alloué.                                                                    */
/*                                                                                                                        */
/* Variable(s) locale(s):    tab - Pointeur de tableau de pointeurs de jours.                                             */
/*-------------------------------------------------------------------------------------------------------------------------*/



jour_t ** AllocationListeContigue()
{
        jour_t ** tab = (jour_t **)malloc(TAILLE_MAX *sizeof(jour_t *));
        return tab;
}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* AllocationJour   Allocation d'un bloc jour.                                                                       */
/*                                                                                                                   */
/* En entrée:    Rien en entrée.                                                                                     */
/*                                                                                                                   */
/* En sortie:    Retourne l'adresse du bloc alloué.                                                                  */
/*                                                                                                                   */
/* Variable(s) locale(s):    pjour - Pointeur sur le bloc alloué.                                                    */
/*-------------------------------------------------------------------------------------------------------------------*/



jour_t * AllocationJour()
{
        our_t * pjour = (jour_t *)malloc(sizeof(jour_t));
        return pjour;
}


/*-------------------------------------------------------------------------------------------------------------------*/
/* LectureSemaineJour   Permet de lire au clavier l'année, la semaine, le jour, l'heure de l'action                  */
/*                              qui sera supprimer et insère ces données année, semaine, jour, heure                 */
/*                               dans un bloc jour.                                                                  */
/*                                                                                                                   */
/* En entrée:    pjour - Pointeur sur le bloc alloué.                                                                */
/*           PcodeLecture - Pointeur sur la case qui contient 1 si la lecture c'est bien passé et 0 sinon*/
/*                                                                                                                   */
/* En sortie:    PcodeLecture - Pointeur sur la case contient 1 si la lecture c'est bien passé et 0 sinon*/
/*-------------------------------------------------------------------------------------------------------------------*/
```

```
void LectureSemaineJour(jour_t * pjour, int * PcodeLecture)
{
	printf("Le contenu de la sauvegarde avant supression\n");
	AfficheSauvegarde();	/*Affiche ce qui était sauvegarder avant la suppression d'une action*/
	 do
	{
		 printf("Veuillez entrer l'année et la semaine\n");
		*PcodeLecture = scanf("%s", pjour->ann_sem);
		if (*PcodeLecture)				/*si la lecture est à marcher*/
		{
			printf("Veuillez entrer le jour et l'heure\n");
			*PcodeLecture = scanf("%s", pjour->jour_heure);		/*j'effectue la deuxième lecture*/
		}
	}while ((!*PcodeLecture) || ((strlen(pjour->ann_sem) != TAILLE_SEMAINE-1) || (strlen(pjour->jour_heure) != TAILLE_JOUR_HR-1)));		/*tantque les chaines lu ne sont pas bonne*/
	if(*PcodeLecture)
	{
		*PcodeLecture = 1;
	}
	else
	{
		*PcodeLecture = 0;
	}
}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* CreerJour        Creer un bloc jour connaissant l'année, la semaine, le jour et l'heure.                          */
/*                                                                                                                   */
/* En entrée:       pjour - Pointeur sur le bloc jour.                                                               */
/*                  psemaine - Pointeur sur la seamine.                                                              */
/*                     paction - Pointeur sur l'action.                                                              */
/*                                                                                                                   */
/* En sortie:       pjour - Pointeur sur le bloc jour.                                                               */
/*-------------------------------------------------------------------------------------------------------------------*/


void CreerJour(jour_t * pjour, semaine_t * psemaine, action_t * paction)
{
        strcpy(pjour->ann_sem, psemaine->ann_sem);
        strcpy(pjour->jour_heure, paction->jour_heure);
}


/*-------------------------------------------------------------------------------------------------------------------*/
/* InsererJour      Insère un bloc jour dans la liste contigue des pointeur de bloc jour à indice donné*/
/*                                                                                                                   */
/* En entrée:   pdebut - Pointeur de début de liste de contigue de pointeurs de jour.                                */
/*                     pjour - Pointeur sur le bloc jour.                                                            */
/*                     indice - L'indice à la qu'elle je dois insérer dans la liste contigue.                        */
/*                                                                                                                   */
/* En sortie:    Rien en sortie.                                                                                     */
/*-------------------------------------------------------------------------------------------------------------------*/
```

```
void InsererJour(jour_t ** pdebut,jour_t * pjour,int indice)
{
        pdebut[indice] = pjour;
}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* LectureMotif         Permet de lire au clavier d'un motif.                                                        */
/*                                                                                                                   */
/* En entrée:    ppmotif - Pointeur de pointeur sur le motif.                                                        */
/*            PcodeLecture - Pointeur sur la case contenant le code de lecture c'est-à-dire 1 si la                  /
/*                                  lecture c'est bien passé et 0 sinon.                                             */
/*                                                                                                                   */
/* En sortie:    ppmotif - Pointeur de pointeur sur le motif.                                                        */
/*            PcodeLecture - Pointeur sur la case contenant le code de lecture c'est-à-dire 1 si la                  */
 /*                                  lecture c'est bien passé et 0 sinon.                                             */
/*                                                                                                                   */
/* Variable(s) locale(s):    TailleMotif - La taille du motif.                                                       */
/*-------------------------------------------------------------------------------------------------------------------*/
```

```
void LectureMotif(char ** ppmotif, int * PcodeLecture)
{
	int TailleMotif;
	printf("Veuillez entrer la taille du motif!\n");
	*PcodeLecture = scanf("%d",&TailleMotif);
	if (*PcodeLecture)				/*si la lecture bien passé*/
	{
		*ppmotif = malloc((TailleMotif+1) * sizeof(char));
		if(*ppmotif != NULL)
		{
			printf("Veuillez entrer votre motif!\n");
			if(fgets(*ppmotif, TailleMotif+1,stdin) != NULL)
			{
			*PcodeLecture = 1;
			}
			else
			{
				*PcodeLecture = 0;
			}
		}
	}
		*
}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* CreerListeJour       Créer la liste contigue de pointeur vers chaque jour dont l'action contenez un */
/*                      motif.                                                                       */
/*                                                                                                   */
/* En entrée:    PteteListe - Pointeur de tete de liste chainée des semaines.                        */
/*                      motif - Chaine caractères qui représente le motif.                           */
/*                    pdebut - Pointeur de début de liste de contigue de pointeurs de jour.          */
/*                     ppfin - L'adresse du pointeur de fin de liste contigue dont les éléments sont */
/*                             des pointeurs de jours.                                               */
/*                                                                                                   */
/* En sortie:       ppfin - L'adresse du pointeur de fin de liste contigue dont les éléments sont    */
/*                          des pointeurs de jours.                                                  */
/*                                                                                                   */
/* Variable(s) locale(s):     NbElement - Le nombre d'éléments déjà insérer dans la liste contigue.  */
/*                            MotifTrouver - Contient la taille du motif si le motif à été trouvé et 0 si le */
/*                                           motif n'a pas été trouvé.                               */
/*                            TailleMotif - La taille du motif.                                      */
/*-------------------------------------------------------------------------------------------------------------------*/
```

```
void CreerListeJour(semaine_t * PteteListe, char * motif, jour_t ** pdebut, jour_t *** ppfin)
{
        int NbElement = 0, MotifTrouver = 0, TailleMotif = strlen(motif);
        semaine_t * pcour1 = PteteListe;            /*initialise à la tete de liste des semaines*/
        action_t * pcour2 = NULL;
        jour_t * pjour = NULL;
        *ppfin = NULL;
        while ((NbElement < TAILLE_MAX) && (pcour1 != NULL))   /*tanque je peux insérer dans ma liste contigue et je suis dans la liste des semaines*/
        {
                pcour2 = pcour1->PlisteAction;              /*initialise à la tete de liste des actions associé à cette semaine*/
                while((NbElement < TAILLE_MAX) && (pcour2 != NULL))/*tanque je peux insérer dans ma liste contigue et je suis dans la liste des actions*/
                {
                        MotifTrouver = RechercherMotif(pcour2->nom_action,motif); /*recherche le motif dans l'action courante*/
                        if(MotifTrouver == TailleMotif)
                        {
                                pjour = AllocationJour();
                                if (pjour != NULL)
                                {
                                        CreerJour(pjour, pcour1, pcour2);
                                        if(*ppfin != NULL)              /*si j'ai insérer dans la liste contigue au moins un élément*/
                                        {
                                                *ppfin = *ppfin + 1;
                                        }
                                        else     /* si j'insère le premier élément de la liste contigue*/
```

```
                                        {
                                                *ppfin = pdebut;
                                         }
                                        InsererJour(pdebut, pjour, NbElement);          /* insère le
jour dans la liste contigue*/
                                        ++NbElement;
                                    }
                                }
                                 pcour2 = pcour2->paction_suiv;                 /*passe à l'action suivante*/
                            }
                            pcour1 = pcour1->psem_suiv;                         /*passe à la seamines suivante*/
                    }
}
```

```
/*-------------------------------------------------------------------------------------------------------------------*/
/* AfficherListeContigue        Affiche la liste contigue de jours.                                                  */
/*                                                                                                                   */
/* En entrée:   pdebut - Pointeur de début de liste de contigue de pointeurs de jour.                                */
/*                       pfin - Pointeur de fin de liste contigue dont les éléments sont des pointeurs de            */
/*                              jours.                                                                               */
/*                                                                                                                   */
/* En sortie:   Rien en sortie.                                                                                      */
/*                                                                                                                   */
/* Variable(s) locale(s):    i, j - Varieble servant d'indice dans le tableau.                                       */
/*                       prec - Pointeur de pointeur de l'élément précédent.                                         */
/*-------------------------------------------------------------------------------------------------------------------*/
```

```
void AfficherListeContigue(jour_t ** pdebut, jour_t ** pfin)

{

	int i=0, j=0;

	jour_t ** prec=NULL;

	if (pfin != NULL)                                   /*si la liste est non vide*/

	{

		prec = pdebut;                          /*initialise le précédent au premier élément*/

		printf("\nLes jours et heures associés à chaque semaine des actions ayant se motifs sont:\n");

		printf("%s\n", pdebut[i]->ann_sem);         /*affiche l'année et semaine courante*/

		while ((pdebut + i) <= pfin)                                    /*tanque qu'on dans la liste contigue*/

		{

			while(((pdebut + i) <= pfin) && !strcmp(pdebut[i]->ann_sem, prec[j]->ann_sem))/*tantque qu'on est dans la liste contigue et que l'année et la semaine du précédent et courant sont égaux*/

			{

				printf("\t%s\n", pdebut[i]->jour_heure);       /*afficher juste le jour et l'heure*/

				++i;

			}

			If (((pdebut + i) <= pfin) && strcmp(pdebut[i]->ann_sem, prec[j]->ann_sem)) /*si on est dans la liste contigue et que l'année et la semaine du précédent et courant sont différents*/

			{

				printf("\n%s\n", pdebut[i]->ann_sem);                   /*affiche l'année et la semaine courante*/

				printf("\t%s\n", pdebut[i]->jour_heure);                 /*affiche le jour et l'heure*/

				j=i;

				++i;
```

```
                        }
                    }
                }
            }
```

```
/*-------------------------------------------------------------------------------------------------------------------------*/
/* LibererListeContigue         Libère la liste contigue.                                                                  */
/*                                                                                                                         */
/* En entrée:    ppdebut - L'adresse du pointeur de tete de liste contigue.                                                */
/*                         ppfin - L'adresse du pointeur de fin de liste contigue.                                         */
/*                                                                                                                         */
/* En sortie:    ppdebut - L'adresse du pointeur de tete de liste contigue.                                                */
/*                         ppfin - L'adresse du pointeur de fin de liste contigue.                                         */
/*                                                                                                                         */
/* Variable(s) locale(s):    i - Variable servant d'indicage.                                                              */
/*-------------------------------------------------------------------------------------------------------------------------*/
```

```
void LibererListeContigue(jour_t *** ppdebut, jour_t *** ppfin)

{

	int i=0;

	if(*ppfin != NULL)				/*si liste non vide*/

	{

		while ((*ppdebut+i) < *ppfin)			/*tantque je ne suis pas à la fin de la liste*/

		{

			 free((*ppdebut[i]));

			++i;

		}

		free((*ppdebut)[i]);

	}

	free(*ppdebut);

	*ppdebut = NULL;

	 *ppfin = NULL;

}
```

Le fichiers main.c qui contient notre fonction principal faisant appel autres fonctions citez plu haut est comme suit:

```
/*-------------------------------------------------------------------------------------------------------------------------*/
/*                                                  Main                                                                   */
/*                                                                                                                         */
/* Role:   Fichier de notre fonction principal faisant appelle à tout les toutes les fonctions                             */
/*           nécéssaires au traiment en ce qui concern:                                                                    */
/*                                                                                                                         */
/*        -Création d'une liste chaine à deux niveaux triée à partir de la lecture dans un fichier.                        */
/*                                                                                                                         */
/*        -Sauvegarde dans un fichier la liste chainée et création d'une liste contigue de bloc de jour                    */
/*         ayant comme anée, semaine, jour, heure de l'action contenant le motif rechercher.                               */
/*                                                                                                                         */
/*           -Suppression d'une action connaissant le l'anée, la semaine, le jour et l'heure.                              */
/*-------------------------------------------------------------------------------------------------------------------------*/



#include "./listeContigue.h"




int main (int argc, char ** argv)
{
        int choix, CodeLecture = 1, CodeSauvegarde = 1;
        char * pmotif = NULL;
        semaine_t * PteteAgenda = NULL;
        jour_t ** PteteListeContigue = NULL, ** PfinListeContigue=NULL, * pjour = NULL;
        if (argc == 2)
```

```
{
        LectureFichier(argv[1], &PteteAgenda, &CodeLecture);
        printf("Veuillez choisir l'opération à faire!\n");
        printf("1. Creer la liste à deux niveaux et sauvegarder\n");
        printf("2. Creer la liste contigue des jours des actions contenant un motif\n");
        printf("3. Supprimer une action.\n");
        scanf("%d", &choix);
         switch(choix)
        {
        case 1:
                if (CodeLecture)                /*si lecture à partir fichier entrée en ligne de commande c'est bien passé*/
                {
                        SauvegardeDansFichier(PteteAgenda,              &CodeSauvegarde); /*sauvegarde liste à deux niveau dans un fichier de sauvegarde*/
                }
                 break;
        case 2:
                if (CodeLecture)                        /*si lecture à partir fichier entrée en ligne de commande c'est bien passé*/
                {
                        PteteListeContigue = AllocationListeContigue();
                        CodeLecture = 1;
                        LectureMotif(&pmotif, &CodeLecture);
                        if ((PteteListeContigue != NULL) && CodeLecture)
                        {
                                CreerListeJour(PteteAgenda,     pmotif,     PteteListeContigue, &PfinListeContigue);
                                AfficherListeContigue(PteteListeContigue, PfinListeContigue);
```

```
                                        LibererListeContigue(&PteteListeContigue,
&PfinListeContigue);
                                        free(pmotif);
                                }
                        }
                         break;
                case 3:
                        if (CodeLecture)                        /*si lecture à partir fichier entrée en
ligne de commande c'est bien passé*/
                        {
                                SauvegardeDansFichier(PteteAgenda, &CodeSauvegarde);
                                pjour = AllocationJour();
                                if(pjour != NULL)
                                {
                                        LectureSemaineJour(pjour,  &CodeLecture);       /*lit  année,
semaine, jour, heure de l'action à supprimer*/
                                        if(CodeLecture)             /*si la lecture c'est bien passé*/
                                        {
                                                supression(&PteteAgenda,pjour->ann_sem,     pjour-
>jour_heure);              /*je supprime l'action*/
                                                free(pjour);
                                        }
                                }
                        }
                }
                LibererListe(&PteteAgenda);
        }
        return 0;
}
```

3 Compte rendu d'exécution

3.1 Makefile

```
#compilateur

CC = gcc

#Profileur

VG = valgrind

#Option

CFLAGS = -ansi -pedantic -Wall -Wextra -g -O2
LDFLAGS = -lm

#liste des fichiers objets

OBJ = main.o listeChainee_semaine.o listeChainee_action.o listeContigue.o


#Regle de production de l'exécutable

main : $(OBJ)
    $(CC) -o main $(OBJ) $(CFLAGS) $(LDFLAGS)


#Règle de production des fichiers objets

listeChainee_semaine.o : listeChainee_semaine.c listeChainee_semaine.h
    $(CC) -c listeChainee_semaine.c $(CFLAGS) $(LDFLAGS)

listeChainee_action.o : listeChainee_action.c listeChainee_action.h
    $(CC) -c listeChainee_action.c $(CFLAGS) $(LDFLAGS)

listeContigue.o: listeContigue.c listeContigue.h
    $(CC) -c listeContigue.c $(CFLAGS) $(LDFLAGS)

main.o : main.c listeContigue.h
    $(CC) -c main.c $(CFLAGS) $(LDFLAGS)
```

```
#Regle pour nettoyer

clean :
    rm $(OBJ)
```

3.2 Jeux de test complets

3.2.1 Création de la liste chainée triée

3.2.1.1 Lorsque le fichier est vide

Dans ce cas particulier nous avons utilisé le fichier lecture1.txt comme donnée d'entrée et le fichier sauvegarde.txt comme donnée de sortie.

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ cat lecture1.txt
cedric@cedric-X302LA:~/C/TP1SDD$ make
make: `main' is up to date.
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture1.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
1
cedric@cedric-X302LA:~/C/TP1SDD$ cat sauvegarde.txt
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Exécution avec valgrind**

```
cedric@cedric-X302LA:~/C/TP1SDD$ cat lecture1.txt
cedric@cedric-X302LA:~/C/TP1SDD$ valgrind ./main lecture1.txt
==3118== Memcheck, a memory error detector
==3118== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==3118== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==3118== Command: ./main lecture1.txt
==3118==
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
1
==3118==
==3118== HEAP SUMMARY:
==3118==     in use at exit: 0 bytes in 0 blocks
==3118==   total heap usage: 2 allocs, 2 frees, 1,136 bytes allocated
==3118==
==3118== All heap blocks were freed -- no leaks are possible
==3118==
==3118== For counts of detected and suppressed errors, rerun with: -v
==3118== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.1.2 Lorsque le fichier ne contient qu'une action

Pour cas la donnée entrée est dans le fichier lecture2.txt représenter par cette capture d'écran:

```
2014250214CMDEPROBRA
```

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ make
make: `main' is up to date.
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture2.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
1
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Les données en sortie obtenu sont dans le fichier sauvegarde.txt**

```
2014250214CMDEPROBRA
```

- **L'exécution avec valgrind**

```
cedric@cedric-X302LA:~/C/TP1SDD$ valgrind ./main lecture2.txt
==3284== Memcheck, a memory error detector
==3284== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==3284== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==3284== Command: ./main lecture2.txt
==3284==
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
1
==3284==
==3284== HEAP SUMMARY:
==3284==     in use at exit: 0 bytes in 0 blocks
==3284==   total heap usage: 4 allocs, 4 frees, 1,184 bytes allocated
==3284==
==3284== All heap blocks were freed -- no leaks are possible
==3284==
==3284== For counts of detected and suppressed errors, rerun with: -v
==3284== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.1.3 Lorsque tous les actions sont sans espace

- **Les données d'entrées sont dans le fichiers lecture3.txt comme suit:**

```
20145171 1PETITDEJEN
20145161 0NETTOYAGES
19992851 6CMDEANGLAI
20123431 3PAUSECAFEE
20152241 5TPDEHTMLJS
19991341 0CMDEHISGEO
20175351 4TPDECONCEP
20162521 8TPDEPHYSIQ
20145161 0NETTOYAGES
19992851 6CMDEBASEDO
20113610 8CALCULDIFF
20172531 0CMDECYBRER
20172530 8CMDERESEAU
20123431 3TPDALGORIT
```

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture3.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
1
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Les données de sortie sont dans le fichier sauvegarde.txt**

```
19991341 0CMDEHISGEO
19992851 6CMDEANGLAI
20113610 8CALCULDIFF
20123431 3PAUSECAFEE
20145161 0NETTOYAGES
20145171 1PETITDEJEN
20152241 5TPDEHTMLJS
20162521 8TPDEPHYSIQ
20172530 8CMDERESEAU
20172531 0CMDECYBRER
20175351 4TPDECONCEP
```

- **L'exécution avec valgrind**

```
cedric@cedric-X302LA:~/C/TP1SDD$ valgrind ./main lecture3.txt
==3460== Memcheck, a memory error detector
==3460== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==3460== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==3460== Command: ./main lecture3.txt
==3460==
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
1
==3460==
==3460== HEAP SUMMARY:
==3460==     in use at exit: 0 bytes in 0 blocks
==3460==   total heap usage: 22 allocs, 22 frees, 1,616 bytes allocated
==3460==
==3460== All heap blocks were freed -- no leaks are possible
==3460==
==3460== For counts of detected and suppressed errors, rerun with: -v
==3460== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.1.4 Lorsqu'il certaines actions possèdes des espaces

- **Les données d'entrées sont dans le fichiers lecture3.txt**

```
201451711PETITDEJEN
201451610NETTOYAGES
199928516CM DE ANGL
201234313PAUSECAFEE
201522415TPDEHTMLJS
199913410CMDEHISGEO
201753514TPDECONCEP
201625218CM DE MATH
201451610NETTOYAGES
199928516CMDEBASEDO
201136108CALCULDIFF
201725310TPs DE SDD
201725308CMDERESEAU
201234313TPDALGORIT
```

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture4.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
1
cedric@cedric-X302LA:~/C/TP1SDD$
```

- Les données e sortie sont dans le fichier sauvegarde.txt

```
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
```

- L'exécution avec valgrind

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture4.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
1
cedric@cedric-X302LA:~/C/TP1SDD$ valgrind ./main lecture4.txt
==3574== Memcheck, a memory error detector
==3574== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==3574== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==3574== Command: ./main lecture4.txt
==3574==
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
1
==3574==
==3574== HEAP SUMMARY:
==3574==     in use at exit: 0 bytes in 0 blocks
==3574==   total heap usage: 22 allocs, 22 frees, 1,616 bytes allocated
==3574==
==3574== All heap blocks were freed -- no leaks are possible
==3574==
==3574== For counts of detected and suppressed errors, rerun with: -v
==3574== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.2 Création d'une liste contigüe de jours

Pour les différents cas particulier concernant la création de la liste contigüe des jours nous utiliserons les données d'entrées situé dans le fichier lecture4.txt.

```
201451711PETITDEJEN
201451610NETTOYAGES
199928516CM DE ANGL
201234313PAUSECAFEE
201522415TPDEHTMLJS
199913410CMDEHISGEO
201753514TPDECONCEP
201625218CM DE MATH
201451610NETTOYAGES
199928516CMDEBASEDO
201136108CALCULDIFF
201725310TPs DE SDD
201725308CMDERESEAU
201234313TPDALGORIT
```

3.2.2.1 Lorsque le motif n'existe pas

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture4.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
2
Veuillez entrer la taille du motif!
3
Veuillez entrer votre motif!
RAZ
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Exécution avec valgrind**

```
cedric@cedric-X302LA:~/C/TP1SDD$ valgrind ./main lecture4.txt
==4151== Memcheck, a memory error detector
==4151== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==4151== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==4151== Command: ./main lecture4.txt
==4151==
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
2
Veuillez entrer la taille du motif!
3
Veuillez entrer votre motif!
RAZ
==4151==
==4151== HEAP SUMMARY:
==4151==     in use at exit: 0 bytes in 0 blocks
==4151==   total heap usage: 23 allocs, 23 frees, 1,212 bytes allocated
==4151==
==4151== All heap blocks were freed -- no leaks are possible
==4151==
==4151== For counts of detected and suppressed errors, rerun with: -v
==4151== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.2.2 Lorsque la taille du motif entrée est plus grande que celui de l'action

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture4.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
2
Veuillez entrer la taille du motif!
12
Veuillez entrer votre motif!
TPs DE SDD P
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Exécution avec valgrind**

```
cedric@cedric-X302LA:~/C/TP1SDD$ valgrind ./main lecture4.txt
==2459== Memcheck, a memory error detector
==2459== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==2459== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==2459== Command: ./main lecture4.txt
==2459==
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
2
Veuillez entrer la taille du motif!
12
Veuillez entrer votre motif!
TPs DE SDD P
==2459==
==2459== HEAP SUMMARY:
==2459==     in use at exit: 0 bytes in 0 blocks
==2459==   total heap usage: 23 allocs, 23 frees, 1,221 bytes allocated
==2459==
==2459== All heap blocks were freed -- no leaks are possible
==2459==
==2459== For counts of detected and suppressed errors, rerun with: -v
==2459== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.2.3 Lorsque le motif existe

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture4.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
2
Veuillez entrer la taille du motif!
2
Veuillez entrer votre motif!
TP

Les jours et heures associés à chaque semaine des actions ayant se motifs sont:
201522
        415

201725
        310

201753
        514
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Exécution avec valgrind**

```
cedric@cedric-X302LA:~/C/TP1SDD$ valgrind ./main lecture4.txt
==2496== Memcheck, a memory error detector
==2496== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==2496== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==2496== Command: ./main lecture4.txt
==2496==
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
2
Veuillez entrer la taille du motif!
2
Veuillez entrer votre motif!
TP

Les jours et heures associés à chaque semaine des actions ayant se motifs sont:
201522
        415

201725
        310

201753
        514
==2496==
==2496== HEAP SUMMARY:
==2496==     in use at exit: 0 bytes in 0 blocks
==2496==   total heap usage: 26 allocs, 26 frees, 1,244 bytes allocated
==2496==
==2496== All heap blocks were freed -- no leaks are possible
==2496==
==2496== For counts of detected and suppressed errors, rerun with: -v
==2496== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.3 Suppression d'une action dans la liste chainée à deux niveaux

Le fichier lecture4.txt tout comme précédemment contiendra les données d'entrées pour cette illustration qui comme suit:

```
201451711PETITDEJEN
201451610NETTOYAGES
199928516CM DE ANGL
201234313PAUSECAFEE
201522415TPDEHTMLJS
199913410CMDEHISGEO
201753514TPDECONCEP
201625218CM DE MATH
201451610NETTOYAGES
199928516CMDEBASEDO
201136108CALCULDIFF
201725310TPs DE SDD
201725308CMDERESEAU
201234313TPDALGORIT
```

3.2.3.1 Suppression au début

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture4.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
3
Le contenu de la sauvegarde avant supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
Veuillez entrer l'année et la semaine
199913
Veuillez entrer le jour et l'heure
410
Le contenu de la sauvegarde après supression
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Exécution avec valgrind**

```
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
3
Le contenu de la sauvegarde avant supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
Veuillez entrer l'année et la semaine
199913
Veuillez entrer le jour et l'heure
410
Le contenu de la sauvegarde après supression
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
==3775==
==3775== HEAP SUMMARY:
==3775==     in use at exit: 0 bytes in 0 blocks
==3775==   total heap usage: 26 allocs, 26 frees, 3,331 bytes allocated
==3775==
==3775== All heap blocks were freed -- no leaks are possible
==3775==
==3775== For counts of detected and suppressed errors, rerun with: -v
==3775== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.3.2 Suppression au milieu

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture4.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
3
Le contenu de la sauvegarde avant supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
Veuillez entrer l'année et la semaine
201451
Veuillez entrer le jour et l'heure
711
Le contenu de la sauvegarde après supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Exécution avec valgrind**

```
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
3
Le contenu de la sauvegarde avant supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
Veuillez entrer l'année et la semaine
201451
Veuillez entrer le jour et l'heure
711
Le contenu de la sauvegarde après supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
==3840==
==3840== HEAP SUMMARY:
==3840==     in use at exit: 0 bytes in 0 blocks
==3840==   total heap usage: 26 allocs, 26 frees, 3,331 bytes allocated
==3840==
==3840== All heap blocks were freed -- no leaks are possible
==3840==
==3840== For counts of detected and suppressed errors, rerun with: -v
==3840== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.3.3 Suppression à la fin

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture4.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
3
Le contenu de la sauvegarde avant supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
Veuillez entrer l'année et la semaine
201753
Veuillez entrer le jour et l'heure
514
Le contenu de la sauvegarde après supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Exécution avec valgrind**

```
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
3
Le contenu de la sauvegarde avant supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
Veuillez entrer l'année et la semaine
201753
Veuillez entrer le jour et l'heure
514
Le contenu de la sauvegarde après supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
==3889==
==3889== HEAP SUMMARY:
==3889==     in use at exit: 0 bytes in 0 blocks
==3889==   total heap usage: 26 allocs, 26 frees, 3,331 bytes allocated
==3889==
==3889== All heap blocks were freed -- no leaks are possible
==3889==
==3889== For counts of detected and suppressed errors, rerun with: -v
==3889== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```

3.2.3.4 Suppression d'une action étant la seule dans ça liste d'une semaine

- **Résultat de l'exécution**

```
cedric@cedric-X302LA:~/C/TP1SDD$ ./main lecture4.txt
Veuillez choisir l'opération à faire!
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
3
Le contenu de la sauvegarde avant supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
Veuillez entrer l'année et la semaine
201234
Veuillez entrer le jour et l'heure
313
Le contenu de la sauvegarde après supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
cedric@cedric-X302LA:~/C/TP1SDD$
```

- **Exécution avec valgrind**

```
1. Creer la liste à deux niveaux et sauvegarder
2. Creer la liste contigue des jours des actions contenant un motif
3. Supprimer une action.
3
Le contenu de la sauvegarde avant supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201234313PAUSECAFEE
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
Veuillez entrer l'année et la semaine
201234
Veuillez entrer le jour et l'heure
313
Le contenu de la sauvegarde après supression
199913410CMDEHISGEO
199928516CM DE ANGL
201136108CALCULDIFF
201451610NETTOYAGES
201451711PETITDEJEN
201522415TPDEHTMLJS
201625218CM DE MATH
201725308CMDERESEAU
201725310TPs DE SDD
201753514TPDECONCEP
==3936==
==3936== HEAP SUMMARY:
==3936==     in use at exit: 0 bytes in 0 blocks
==3936==   total heap usage: 26 allocs, 26 frees, 3,331 bytes allocated
==3936==
==3936== All heap blocks were freed -- no leaks are possible
==3936==
==3936== For counts of detected and suppressed errors, rerun with: -v
==3936== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cedric@cedric-X302LA:~/C/TP1SDD$
```