Compte rendu du TP1 de Structure de Données

Laurent Valentin, Malrin Vincent

30 mars 2013

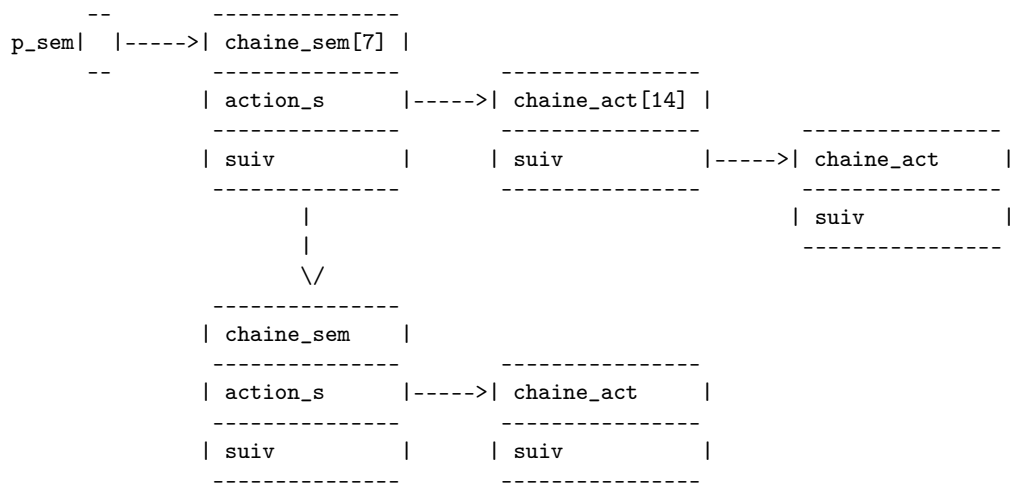
Table des matières

1.1 Objet du TP

Création d'un agenda grâce à une liste chaînée à deux niveaux.

1.2 Structure de données

Chaque bloc du premier niveau contient une année, le numéro de la semaine, un pointeur vers la liste des actions de la semaine et un pointeur vers la semaine suivante. Les blocs du second niveau (liste chainée des actions) contiennent le jour de la semaine, l'heure, le nom de l'action et le pointeur vers l'action suivante.



1.3 Organisation du code source

agenda.h

C'est le fichier d'entête.

insertion.c

Ce fichier contient les fonctions :

- InsererSem
- InsererAct
- Liberation

recherche.c

Ce fichier contient les fonctions :

- RechercheSem
- RechercheAct

suppression.c

Ce fichier contient les fonctions :

- SuppressionAct
- SuppressionSem

afficher.c

Ce fichier contient les fonctions :

- Affichage
- Menu

bilatere.c

Ce fichier contient les fonctions :

- TransfoBilatere
- IsertionBil
- LiberationBil

ecriture.c

Ce fichier contient les fonctions :

- Ecriture
- Lecture

recherche_motif.c

Ce fichier contient les fonctions :

- RechercheMotif
- AffichageMotif

1.4 Codes Sources

1.4.1 EN TÊTE : agenda.h

```c
/* Malrin Vincent
 * Laurent Valentin
 * 25/03/2013
 * EN TETE : agenda.h
 */

/*GARDIEN*/
#ifndef AGENDA_H
#define AGENDA_H

/*DEFINE*/
#define TLIBELLE 10
#define TACT (TLIBELLE + 3)
#define TMAX 100

/* INCLUDE */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*STRUCTURES*/
typedef struct struct2
{
    char              chaine_act[3 + TLIBELLE + 1];
    struct struct2 *  suiv;
} action_t;

typedef struct struct1
{
    char              chaine_sem[7];
    action_t *        action_s;
    struct struct1 *  suiv;
} semaine_t;

typedef struct struct3
{
    char              motif[13];
} motif_t;

typedef struct struct4
{
    char              chaine_sem[7];
    action_t *        action_s;
    struct struct4 *  prec;
    struct struct4 *  suiv;
} bilatere_t;

/*-----------------------------------------------------*/
/*           INSERTION (insertion.c)            */
/*-----------------------------------------------------*/


/* FONCTION : InsererSem
 * Insertion d'une nouvelle semaine dans la liste chainee.
 *
```

```
 * entree : le pointeur de recherche (semaine_t ** prec)
 *         les informations à inserer (char * chaine_sem)
 * varloc : le nouveau pointeur à inserer (semaine_t * new)
 * sortie : le nouveau pointeur (semaine_t * new)
 */

semaine_t * InsererSem(semaine_t ** prec, char * chaine_sem);

/* FONCTION : InsererAct
 * Insertion d'une nouvelle action dans la liste chainee.
 *
 * entree : le pointeur de recherche (action_t ** prec)
 *         les informations à inserer (char * chaine_act)
 * varloc : le nouveau pointeur à inserer (action_t * new)
 * sortie : le nouveau pointeur (action_t * new)
 */

action_t * InsererAct(action_t ** prec, char * chaine_act);

/* FONCTION : Liberation
 * Libere toute les données allouees.
 *
 * entree : le pointeur de debut de la liste chainee (semaine_t * p_sem)
 * varloc : les pointeurs courants (action_t * cour_act et semaine_t * cour_sem)
 *         les pointeurs temporaires (action_t * tmp_act et semaine_t * tmp_sem)
 * sortie : rien
 */

void Liberation(semaine_t * p_sem);

/*-------------------------------------------------*/
/*           RECHERCHE (recherche.c)               */
/*-------------------------------------------------*/

/* FONCTION : RechercheSem
 * Recherche de la semaine dans la liste chainee.
 *
 * entree : le pointeur de debut de la liste chainee (semaine_t ** p_sem)
 *         les informations de à rechercher (char * chaine_sem)
 * varloc : le pointeur precedent (semaine_t ** prec)
 *         le pointeur courant (semaine_t * cour)
 * sortie : le pointeur correspondant à la recherche (semaine_t ** prec)
 */

semaine_t ** RechercheSem(semaine_t ** p_sem, char * chaine_sem);

/* FONCTION : RechercheAct
 * Recherche de l'action dans la liste chainee.
 *
 * entree : le pointeur de debut de la liste chainée (action_t ** p_act)
 *         les informations de à rechercher (char * chaine_act)
 * varloc : le pointeur precedent (action_t ** prec)
 *         le pointeur courant (action_t * cour)
 * sortie : le pointeur correspondant à la recherche (action_t ** prec)
 */

action_t ** RechercheAct(action_t ** p_act, char * chaine_act);

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

```
/*           SUPPRESSION (suppression.c)         */
/*-------------------------------------------------*/

/* FONCTION : SuppressionAct
 * Suppression de l'action dans la liste chainée.
 *
 * entree : le pointeur de debut de la liste chainee (action_t ** p_act)
 *         les informations à rechercher (char * jour)
 * varloc : le pointeur precedent (action_t ** prec)
 *         le pointeur courant (action_t * cour)
 */

void SuppressionAct(action_t ** prec);

/* FONCTION : SuppressionSem
 * Suppression de la semaine dans la liste chainee.
 *
 * entree : le pointeur de debut de la liste chaiee (semaine_t ** p_sem)
 *         les informations à rechercher (char * jour)
 * varloc : le pointeur precedent (semaine_t ** prec)
 *         le pointeur courant (semaine_t * cour)
 */

void SuppressionSem(semaine_t ** p_sem);

/*-------------------------------------------------*/
/*              AFFICHER (afficher.c)              */
/*-------------------------------------------------*/

/* FONCTION : Affichage
 * Affiche les informations des listes chainees (semaine_t).
 *
 * entree : le pointeur de debut de la liste chainee (semaine_t * p_sem)
 * varloc : le pointeur courant (action_t * cour)
 * sortie : rien
 */

void Affichage(semaine_t * p_sem);

/* FONCTION : Menu
 * Affichage du menu.
 * entree : code de saisie (int saisie)
 *         le ponteur vers le choix (int * choix)
 * sortie : rien
 */

void Menu(int saisie, int * choix);

/*-------------------------------------------------*/
/*              BILATERE (bilatere.c)              */
/*-------------------------------------------------*/

/* FONCTION : Transformation
 * Transforme la liste des semaines en liste bilatere.
 *
 * entree : le pointeur de debut de la liste chainee (semaine_t ** p_sem)
 *         le pointeur de debut de la liste bilatere (semaine_t * p_bil)
 * varloc : le pointeurs precedents (semaine_t ** prec, bilatere_t ** prec_bil)
 *         le pointeur courant de la liste bilatere (bilatere_t * nouv)
```

```
 * sortie : le pointeur de debut de la liste bilatere (semaine_t * p_bil)
 */

bilatere_t * TransfoBilatere (semaine_t ** p_sem, bilatere_t * p_bil);

/* FONCTION : InsertionBil
 * Insère une celulle.
 *
 * entree : l'adresse du pointeur où il faut inserer (bilatere_t ** b)
 *          le pointeur de la cellule à inserer (bilatere_t * elt)
 * sortie : l'adresse de l'élement inséré (semaine_t ** adr_bil)
 */

bilatere_t ** InsertionBil(bilatere_t ** b, bilatere_t * elt);

/* FONCTION : LiberationBil
 * Libere toute les données allouées dans la structure bilatere.
 *
 * entree : le pointeur de debut de la liste bilatere (bilatere_t * p_bil)
 * varloc : les pointeurs courants (action_t * cour_act et bilatere_t * cour_bil)
 *          les pointeurs temporaires (action_t * tmp_act et bilatere_t * tmp_bil)
 * sortie : rien
 */

void LiberationBil(bilatere_t * p_bil);

/*-----------------------------------------------------*/
/*              ECRITURE (ecriture.c)                  */
/*-----------------------------------------------------*/

/* FONCTION : Ecriture
 * Ecriture de la liste dans un fichier texte.
 *
 * entree : le pointeur de debut de liste (semaine_t * p_sem)
 *          le nom du fichier à ouvrir (char * nom)
 * varloc : le nouveau pointeur à inserer (semaine_t * new)
 * sortie : code erreur fourni en parametre de sortie (int code)
 */

int Ecriture(semaine_t * p_sem, char * nom);

/* FONCTION : Lecture
 * Lecture dans un fichier texte et creation de la liste.
 *
 * entree : le pointeur de la structure (semaine_t ** p_sem)
 *          le nom du fichier à ouvrir (char * nom_f)
 * varloc : pointeurs de parcours, les informations à copier
 * sortie : code erreur fourni en parametre de sortie (int erreur)
 */

int Lecture(semaine_t ** p_sem, char * nom_f);

/*-----------------------------------------------------*/
/*  RECHERCHE DE MOTIF (recherche_motif.c)             */
/*-----------------------------------------------------*/

/* FONCTION : RechercheMotif
 * Crée une liste contigue des jours ou une action contenant un motif donné est présente.
 *
```

```
 * entree : pointeur sur la liste chainee des semaines (semaine_t * p_sem).
 *         pointeur sur la liste contigue des jours (motif_t * p_motif).
 *         chaine du motif à rechercher (char * motif_rech).
 * varloc : les pointeurs courants (semaine_t * cour_sem, action_t * cour_act).
 *         compteur de la liste contigue (int compteur).
 *         chaine de copie (char chaine_motif[10], char libelle[11], char dechet[TMAX]).
 * sortie : l'adresse de la dernière cellule de recherche.
 */

int RechercheMotif(semaine_t * p_sem, motif_t * p_motif, char * motif_rech);

/* FONCTION : AffichageMotif
 * Crée une liste contigue
 *
 * entree : pointeur sur la liste chainee des semaines (semaine_t * p_sem).
 *         pointeur sur la liste contigue des jours (motif_t * p_motif).
 *         le nombre de cellules completes (int compteur).
 * varloc : les pointeurs courants (semaine_t * cour_sem, action_t * cour_act).
 *         compteur de la liste contigue (int compteur).
 *         chaine de copie (char chaine_motif[10], char libelle[11], char dechet[TMAX]).
 * sortie : l'adresse de la dernière cellule de recherche.
 */

void AffichageMotif(motif_t * p_motif, int compteur);

#endif
```

1.4.2 MAIN : agenda.c

```c
/* Malrin Vincent
 * Laurent Valentin
 * 25/03/2013
 * MAIN : agenda.c
 * Creer une structure de données en mémoire à partir d'un fichier texte.
 *
 * EXEMPLE : 201321110TP de SDD
 *          chaine_sem[6]chaine_act[3 + 10]
 *
 * entree : choix du menu (int)
 *         nom du fichier texte (char *)
 *         arguments : date (char *) ou nom de fichier d'ecriture (char *)
 *
 * USAGE : agenda <choix menu> <fichier de lecture> <argument>
 */

/*INCLUDE*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "agenda.h"

int main (int argc, char ** argv)
{
    /*variables locales*/
    semaine_t *       p_sem = NULL;

    semaine_t **      rech_sem = NULL;
    action_t **       rech_act = NULL;

    bilatere_t *      p_bil = NULL;

    motif_t *         p_motif = NULL; /*liste de motifs*/
    int               compteur_motif = 0;

    char              chaine_sem[7]; /*copies des informations*/
    char              chaine_act[4];
    char              dechet[100];
    char              libelle[11];

    int               choix = 1; /*choix par defaut*/
    int               erreur = 0;

    if (argc < 3) /*vérification arguments*/
    {
        printf("USAGE : %s <choix menu> <fichier de lecture> <argument>\n",argv[0]);
        /*affichage du menu et saisie du choix*/
        Menu(0,&choix);
    }
    else
    {
        /*initialisation du choix*/
        choix = atoi(argv[1]);

        /*lecture du fichier*/
        erreur = Lecture(&p_sem,argv[2]);

```

```
58          /*initialisation des chaines*/
59          chaine_sem[6] = '\0';
60          chaine_act[3] = '\0';
61
62          while (choix != 0)
63          {
64
65              switch(choix)
66              {
67                  case 0 :
68                      break;
69
70                  case 1 :
71                      /*affichage*/
72                      Affichage(p_sem);
73                      break;
74
75                  case 2 :
76                      /*recherche par date*/
77                      if ((argc >= 4) && (strlen(argv[3]) == 9)) /*vérification arguments*/
78                      {
79                          /*copie de la date à rechercher*/
80                          sscanf(argv[3],"%6c%3c%*c",chaine_sem,chaine_act);
81
82                          /*recherche dans l'agenda*/
83                          rech_sem = RechercheSem(&p_sem,chaine_sem);
84
85                          if (*rech_sem != NULL)
86                          {
87                              rech_act = RechercheAct(&(**rech_sem).action_s,chaine_act);
88                          }
89
90                          if ((*rech_sem != NULL) && (*rech_act != NULL))
91                          {
92                              sscanf((**rech_act).chaine_act,"%3c%10c",dechet,libelle);
93                              libelle[10] = '\0';
94                              printf("LIBELLE RECHERCHEE : %s\n",libelle);
95                          }
96                          else if ((*rech_sem == NULL) || (*rech_act == NULL)) /*affichage de l'
                                erreur*/
97                          {
98                              printf("Aucune page n'est trouvee\n");
99                          }
100                     }
101                     else
102                     {
103                         printf("Erreur : date incomplete\n");
104                     }
105                     break;
106
107                 case 3 :
108                     /*suppression par date*/
109                     if ((argc >= 4) && (strlen(argv[3]) == 9)) /*vérification arguments*/
110                     {
111                         /*copie de la date à supprimer*/
112                         sscanf(argv[3],"%6c%3c%*c",chaine_sem,chaine_act);
113
114                         /*recherche dans l'agenda*/
115                         rech_sem = RechercheSem(&p_sem,chaine_sem);
```

```
116
117                     if (*rech_sem != NULL)
118                     {
119                         rech_act = RechercheAct(&(**rech_sem).action_s,chaine_act);
120                     }
121
122                     if ((*rech_sem != NULL) && (*rech_act != NULL))
123                     {
124                         sscanf((**rech_act).chaine_act,"%3c%10c",dechet,libelle);
125                         libelle[10] = '\0';
126                         printf("LIBELLE SUPPRIMEE : %s\n\n",libelle);
127
128                         SuppressionAct(rech_act);
129
130                         if ((**rech_sem).action_s == NULL)
131                         {
132                             SuppressionSem(rech_sem);
133                         }
134
135                         Affichage(p_sem);
136                     }
137                     else if ((*rech_sem == NULL) || (*rech_act == NULL)) /*affichage de l'
                         erreur*/
138                     {
139                         printf("Aucune page n'est supprimee\n");
140                     }
141                 }
142                 else
143                 {
144                     printf("Erreur : date incomplete\n");
145                 }
146                 break;
147
148             case 4 :
149                 /*ecriture*/
150                 if (argc >= 4) /*vérification arguments*/
151                 {
152                     Ecriture(p_sem,argv[3]);
153                 }
154                 break;
155
156             case 5 :
157                 /*transformation en liste bilatère*/
158                 p_bil = (bilatere_t *) malloc(sizeof(bilatere_t));
159                 strcpy((*p_bil).chaine_sem,"\0");
160                 (*p_bil).action_s = NULL;
161                 (*p_bil).suiv = p_bil;
162                 (*p_bil).prec = p_bil;
163
164                 if (p_bil) /*vérification de l'allocation*/
165                 {
166                     TransfoBilatere(&p_sem,p_bil);
167                     LiberationBil(p_bil);
168                     free(p_bil);
169                 }
170                 break;
171
172             case 6 :
173                 /*recherche de motif*/
```

```
174                     p_motif = (motif_t *) malloc(sizeof(motif_t)*TMAX);
175
176                     if ((argc >= 4) && (strlen(argv[3]) < 11) && p_motif) /*vérification arguments
                            */
177                     {
178                         /*copie du motif à rechercher*/
179                         sscanf(argv[3],"%s%*c",libelle);
180
181                         compteur_motif = RechercheMotif(p_sem,p_motif,libelle);
182                         AffichageMotif(p_motif,compteur_motif);
183                         free(p_motif);
184                     }
185                     else
186                     {
187                         printf("Erreur : motif incorrect\n");
188                     }
189                     break;
190
191                 default :
192                     printf("Choix non identifie !\n");
193                     break;
194             }
195
196             /*reinitialisation des pointeurs*/
197             rech_sem = NULL;
198             rech_act = NULL;
199
200             /*reinitialisation du choix (sinon boucle infinie)*/
201             /*Menu(1,&choix);*/
202             choix = 0;
203         }
204
205         /*libération des ressources*/
206         Liberation(p_sem);
207
208         printf("Fin du programme\n");
209     }
210
211     return 0;
212 }
```

1.4.3 INSERTION : insertion.c

```
1  /* Malrin Vincent
2   * Laurent Valentin
3   * 25/03/2013
4   * INSERTION : insertion.c
5   */
6
7  /*INCLUDE*/
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include <string.h>
11 #include "agenda.h"
12
13 /* FONCTION : InsererSem
14  * Insertion d'une nouvelle semaine dans la liste chainee.
15  *
16  * entree : le pointeur de recherche (semaine_t ** prec)
```

```
 *         les informations à inserer (char * chaine_sem)
 * varloc : le nouveau pointeur à inserer (semaine_t * new)
 * sortie : le nouveau pointeur (semaine_t * new)
 */

semaine_t * InsererSem(semaine_t ** prec, char * chaine_sem)
{
    /*variales locales*/
    semaine_t *       new = (semaine_t *) malloc (sizeof(semaine_t));

    if(new)
    {
        /*copie des informations*/
        strcpy((*new).chaine_sem,chaine_sem);
        (*new).action_s = NULL;
        (*new).suiv = NULL;

        /*insertion du pointeur new*/
        (*new).suiv = *prec;
        *prec = new;
    }
    return new;
}

/* FONCTION : InsererAct
 * Insertion d'une nouvelle action dans la liste chainee.
 *
 * entree : le pointeur de recherche (action_t ** prec)
 *         les informations à inserer (char * chaine_act)
 * varloc : le nouveau pointeur à inserer (action_t * new)
 * sortie : le nouveau pointeur (action_t * new)
 */

action_t * InsererAct(action_t ** prec, char * chaine_act)
{
    /*variales locales*/
    action_t *        new = (action_t *) malloc (sizeof(action_t));

    if(new)
    {
        /*copie des informations*/
        strcpy((*new).chaine_act,chaine_act);

        /*insertion du pointeur new*/
        (*new).suiv = *prec;
        *prec = new;
    }
    return new;
}

/* FONCTION : Liberation
 * Libere toute les données allouees.
 *
 * entree : le pointeur de debut de la liste chainee (semaine_t * p_sem)
 * varloc : les pointeurs courants (action_t * cour_act et semaine_t * cour_sem)
 *         les pointeurs temporaires (action_t * tmp_act et semaine_t * tmp_sem)
 * sortie : rien
 */

```

```c
void Liberation(semaine_t * p_sem)
{
    /*variables locales*/
    semaine_t *       cour_sem = p_sem;
    action_t *        cour_act;
    action_t *        tmp_act;
    semaine_t *       tmp_sem;

    /*parcours*/
    while (cour_sem)
    {

        /*on initialise le pointeur d'action*/
        cour_act = (*cour_sem).action_s;

        while (cour_act)
        {
            tmp_act = cour_act;

            /*on avance les pointeurs de action_t*/
            cour_act = (*cour_act).suiv;

            /*libération de la structure action_t*/
            free(tmp_act);
        }

        tmp_sem = cour_sem;

        /*on avance le pointeur de semaine_t*/
        cour_sem = (*cour_sem).suiv;

        /*libération de la structure semaine_t*/
        free(tmp_sem);

    }

    p_sem = NULL;
}
```

1.4.4 AFFICHAGE : afficher.c

```c
/* Malrin Vincent
 * Laurent Valentin
 * 25/03/2013
 * AFFICHAGE : afficher.c
 */

/*INCLUDE*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "agenda.h"

/* FONCTION : Affichage
 * Affiche les informations des listes chainées (semaine_t).
 *
 * entree : le pointeur de debut de la liste chainée (semaine_t * p_sem)
 * varloc : les pointeurs courants (action_t * cour_act et semaine_t * cour_sem)
 *          les informations à afficher (char)
```

```
 * sortie : rien
 */

void Affichage(semaine_t * p_sem)
{
    /*variables locales*/
    semaine_t *       cour_sem = p_sem;
    action_t *        cour_act;
    char              annee[5];
    char              semaine[3];
    char              jour[2];
    char              heure[3];
    char              libelle[11];

    /*parcours*/
    while (cour_sem)
    {
        /*initialisation des chaines*/
        jour[1] = '\0';
        annee[4] = '\0';
        semaine[2] = '\0';
        heure[2] = '\0';
        libelle[10] = '\0';


        /*on initialise le pointeur d'action*/
        cour_act = (*cour_sem).action_s;

        /*mise à jour des informations à afficher*/
        /*chaine_sem,semaine*/
        sscanf(cour_sem->chaine_sem,"%4c%2c",annee,semaine);

        while (cour_act)
        {
            /*jour,heure,libelle*/
            sscanf(cour_act->chaine_act,"%1c%2c%10c",jour,heure,libelle);

            /*affichage des informations*/
            printf("ANNEE : %s\n",annee);
            printf("SEMAINE : %s\n",semaine);
            printf("JOUR : %s\n",jour);
            printf("HEURE : %s\n",heure);
            printf("LIBELLE : %s\n",libelle);
            printf("\n");

            /*on avance les pointeurs de action_t*/
            cour_act = (*cour_act).suiv;
        }

        /*on avance le pointeur de semaine_t*/
        cour_sem = (*cour_sem).suiv;
    }

    printf("-----------------------------------------\n");
}


/* FONCTION : Menu
 * Affichage du menu.
```

```
 * entree : code de saisie (int saisie)
 *         le ponteur vers le choix (int * choix)
 * sortie : rien
 */

void Menu(int saisie, int * choix)
{
    /*saisie controle activation du scanf*/
    printf("\n<choix menu> :\n");
    printf("0) Quitter\n");
    printf("1) Affichage\n");
    printf("2) Recherche par date\n");
    printf("3) Suppression\n");
    printf("4) Ecriture\n");
    printf("5) Transformation en liste bilatere\n");
    printf("6) Recherche de motifs\n");

    printf("\n<argument> :\n<menu 1,2> -> date\n<menu 5> -> fichier d'ecriture\n<menu 6> -> motif
        \n");

    if (saisie == 1)
    {
        printf("Veuillez choisir votre choix ? ");
        fflush(stdout);
        scanf("%d%*c",choix);
        printf("\n");
    }
}
```

1.4.5 RECHERCHE : recherche.c

```
/* Malrin Vincent
 * Laurent Valentin
 * 25/03/2013
 * RECHERCHE : recherche.c
 */

/*INCLUDE*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "agenda.h"

/* FONCTION : RechercheSem
 * Recherche de la semaine dans la liste chainee.
 *
 * entree : le pointeur de debut de la liste chainee (semaine_t ** p_sem)
 *         les informations de à rechercher (char * chaine_sem)
 * varloc : le pointeur precedent (semaine_t ** prec)
 *         le pointeur courant (semaine_t * cour)
 * sortie : le pointeur correspondant à la recherche (semaine_t ** prec)
 */

semaine_t ** RechercheSem(semaine_t ** p_sem, char * chaine_sem)
{
    /*variables locales*/
    char              annee[5];
    char              cour_annee[5];
    char              semaine[3];
```

```
29      char              cour_semaine[3];
30      semaine_t **      prec = p_sem;
31      semaine_t *       cour = *p_sem;
32
33      if (cour != NULL) /*si la liste des semaines est nulle*/
34      {
35          /*initialisation des chaines*/
36          sscanf(chaine_sem,"%4c%2c",annee,semaine);
37          sscanf(cour->chaine_sem,"%4c%2c",cour_annee,cour_semaine);
38          annee[4] = '\0';
39          cour_annee[4] = '\0';
40          semaine[2] = '\0';
41          cour_semaine[2] = '\0';
42      }
43
44      /*parcours*/
45      while ((cour != NULL) && ((strcmp(cour_annee,annee) < 0) || ((strcmp(cour_annee,annee) == 0)
            && (strcmp(cour_semaine,semaine) < 0))))
46      {
47          /*on avance les pointeurs*/
48          prec = &(*cour).suiv;
49          cour = (*cour).suiv;
50
51          /*si le pointeur suivant est vide, la copie est impossible*/
52          if (cour != NULL)
53          {
54              sscanf(cour->chaine_sem,"%4c%2c",cour_annee,cour_semaine);
55          }
56      }
57      if ((cour != NULL) && ((strcmp(cour_annee,annee) > 0) || (strcmp(cour_semaine,semaine) > 0)))
58      {
59          *prec = NULL; /*la date est anterieure aux dates de listes*/
60      }
61
62      return prec;
63  }
64
65  /* FONCTION : RechercheAct
66   * Recherche de l'action dans la liste chainee.
67   *
68   * entree : le pointeur de debut de la liste chainée (action_t ** p_act)
69   *          les informations de à rechercher (char * chaine_act)
70   * varloc : le pointeur precedent (action_t ** prec)
71   *          le pointeur courant (action_t * cour)
72   * sortie : le pointeur correspondant à la recherche (action_t ** prec)
73  */
74
75  action_t ** RechercheAct(action_t ** p_act, char * chaine_act)
76  {
77      /*variables locales*/
78      char              jour[2];
79      char              cour_jour[2];
80      char              heure[3];
81      char              cour_heure[3];
82      action_t **       prec = p_act;
83      action_t *        cour = *p_act;
84
85      if (cour != NULL) /*si la liste des actions est nulle*/
86      {
```

```
87          /*initialisation des chaines*/
88          sscanf(chaine_act,"%1c%2c",jour,heure);
89          sscanf(cour->chaine_act,"%1c%2c",cour_jour,cour_heure);
90          jour[1] = '\0';
91          cour_jour[1] = '\0';
92          heure[2] = '\0';
93          cour_heure[2] = '\0';
94      }
95
96      /*parcours*/
97      while ((cour != NULL) && ((strcmp(cour_jour,jour) < 0) || ((strcmp(cour_jour,jour) == 0) && (
            strcmp(cour_heure,heure) < 0))))
98      {
99          /*on avance les pointeurs*/
100         prec = &(*cour).suiv;
101         cour = (*cour).suiv;
102
103         /*si le pointeur suivant est vide, la copie est impossible*/
104         if (cour != NULL)
105         {
106             sscanf(cour->chaine_act,"%1c%2c",cour_jour,cour_heure);
107         }
108     }
109
110     if ((cour != NULL) && ((strcmp(cour_jour,jour) > 0) || (strcmp(cour_heure,heure) > 0)))
111     {
112         *prec = NULL; /*la date est anterieure aux dates de listes*/
113     }
114     return prec;
115 }
```

1.4.6 SUPPRESSION : suppression.c

```
1  /* Malrin Vincent
2   * Laurent Valentin
3   * 25/03/2013
4   * SUPPRESSION : suppression.c
5   */
6
7  /*INCLUDE*/
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include <string.h>
11 #include "agenda.h"
12
13
14 /* FONCTION : SuppressionAct
15  * Suppression de la semaine dans la liste chainée.
16  *
17  * entree : l'adresse de la structure à supprimer (action_t ** prec)
18  * varloc : le pointeur temporaire (action_t * tmp)
19  */
20
21 void SuppressionAct(action_t ** prec)
22 {
23     /*variables locales*/
24     action_t *        tmp = *prec;
25
26     /*le pointeur à supprimer est NULL*/
```

```
    if (*prec == NULL)
    {
        printf("Erreur\n");
    }
    else
    {
        *prec = ((**prec).suiv);
        free(tmp);
    }
}

/* FONCTION : SuppressionSem
 * Suppression de la semaine dans la liste chainée.
 *
 * entree : l'adresse de la structure à supprimer (semaine_t ** prec)
 * varloc : le pointeur temporaire (semaine_t * tmp)
 */

void SuppressionSem(semaine_t ** prec)
{
    /*variables locales*/
    semaine_t *       tmp = *prec;

    /*le pointeur à supprimer est NULL*/
    if (*prec == NULL)
    {
        printf("Erreur\n");
    }
    else
    {
        *prec = (**prec).suiv;
        free(tmp);
    }
}
```

1.4.7 RECHERCHE DE MOTIF : motif.c

```
/* Malrin Vincent
 * Laurent Valentin
 * 25/03/2013
 * RECHERCHE DE MOTIF : motif.c
 */

/*INCLUDE*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "agenda.h"

/* FONCTION : RechercheMotif
 * Crée une liste contigue des jours ou une action contenant un motif donné est présente.
 *
 * entrée : pointeur sur la liste chainée des semaines (semaine_t * p_sem).
 *          pointeur sur la liste contigue des jours (motif_t * p_motif).
 *          chaine du motif à rechercher (char * motif_rech).
 * varloc : les pointeurs courants (semaine_t * cour_sem, action_t * cour_act).
 *          compteur de la liste contigue (int compteur).
 *          chaine de copie (char chaine_motif[10], char libelle[11], char dechet[TMAX]).
 * sortie : l'adresse de la dernière cellule de recherche.
```

```c
 */

int RechercheMotif(semaine_t * p_sem, motif_t * p_motif, char * motif_rech)
{
    /*variables locales*/
    semaine_t *       cour_sem = p_sem;
    action_t *        cour_act = (*cour_sem).action_s; /*on initialise le pointeur d'action*/
    int               compteur = 0;

    char              dechet[TMAX];
    char              libelle[11];

    /*compteur*/
    while (cour_sem)
    {
        /*initialisation des chaines*/
        libelle[11] = '\0';

        /*on initialise le pointeur d'action*/
        cour_act = (*cour_sem).action_s;

        while (cour_act)
        {
            sscanf(cour_act->chaine_act,"%3c%10c",dechet,libelle);
            if (strstr(libelle,motif_rech) != NULL)
            {
                /*copie de la date du motif*/
                strcpy(p_motif[compteur].motif,cour_sem->chaine_sem);
                strncat(p_motif[compteur].motif,cour_act->chaine_act,3);
                p_motif[compteur].motif[10] = '\0';
                compteur++;
            }

            /*on avance les pointeurs de action_t*/
            cour_act = (*cour_act).suiv;
        }

        /*on avance le pointeur de semaine_t*/
        cour_sem = (*cour_sem).suiv;
    }
    return compteur;
}

/* FONCTION : AffichageMotif
 * Crée une liste contigue
 *
 * entrée : pointeur sur la liste chainée des semaines (semaine_t * p_sem).
 *          pointeur sur la liste contigue des jours (motif_t * p_motif).
 *          le nombre de cellules complètes (int compteur).
 * varloc : les pointeurs courants (semaine_t * cour_sem, action_t * cour_act).
 *          compteur de la liste contigue (int compteur).
 *          chaine de copie (char chaine_motif[10], char libelle[11], char dechet[TMAX]).
 * sortie : l'adresse de la dernière cellule de recherche.
 */

void AffichageMotif(motif_t * p_motif, int compteur)
{
    /*variables locales*/
    int               compteur_aff = 0;
```

```
82
83      if (compteur == 0) /*cas d'une liste vide*/
84      {
85          printf("Aucune date correspondante\n");
86      }
87      else
88      {
89          /*parcours*/
90          while (compteur_aff < compteur)
91          {
92              printf("%s\n",p_motif[compteur_aff].motif);
93              compteur_aff++;
94          }
95      }
96  }
```

1.4.8 ECRITURE : ecriture.c

```
1   /* Malrin Vincent
2    * Laurent Valentin
3    * 25/03/2013
4    * ECRITURE : ecriture.c
5    */
6
7   /*INCLUDE*/
8   #include <stdio.h>
9   #include <stdlib.h>
10  #include <string.h>
11  #include "agenda.h"
12
13  /* FONCTION : Ecriture
14   * Ecriture de la liste dans un fichier texte.
15   *
16   * entree : le pointeur de debut de liste (semaine_t * p_sem)
17   *          le nom du fichier à ouvrir (char * nom)
18   * varloc : le nouveau pointeur à inserer (semaine_t * new)
19   * sortie : code erreur fourni en parametre de sortie (int code)
20   */
21
22  int Ecriture(semaine_t * p_sem, char * nom)
23  {
24      /*variables locales*/
25      FILE *            f_ecriture = fopen(nom, "w");
26      semaine_t *       cour_sem = p_sem;
27      action_t *        cour_act;
28      char              chaine_sem[7];
29      char              chaine_act[3 + TLIBELLE + 1];
30      int               code = 0;
31
32      /*parcours*/
33      if (!f_ecriture)
34      {
35          printf("Ecriture impossible !");
36          code = 1;
37      }
38      else
39      {
40          while (cour_sem)
41          {
```

```
42              /*initialisation des chaines*/
43              chaine_sem[6] = '\0';
44              chaine_act[TACT] = '\0';
45
46              /*on initialise le pointeur d'action*/
47              cour_act = (*cour_sem).action_s;
48
49              /*mise à jour des informations à afficher*/
50              /*annee,semaine*/
51              strcpy(chaine_sem,cour_sem->chaine_sem);
52
53              while (cour_act)
54              {
55                   /*jour,heure,libelle*/
56                  strcpy(chaine_act,cour_act->chaine_act);
57
58                  /*affichage des informations*/
59                  fputs(chaine_sem,f_ecriture);
60                  fputs(chaine_act,f_ecriture);
61                  fputc('\n',f_ecriture);
62
63                  /*on avance les pointeurs de action_t*/
64                  cour_act = (*cour_act).suiv;
65              }
66
67              /*on avance le pointeur de semaine_t*/
68              cour_sem = (*cour_sem).suiv;
69          }
70
71          /*fermeture du fichier d'ecriture*/
72          fclose(f_ecriture);
73      }
74
75      return code;
76  }
77
78  /* FONCTION : Lecture
79   * Lecture dans un fichier texte et creation de la liste.
80   *
81   * entree : le pointeur de la structure (semaine_t ** p_sem)
82   *          le nom du fichier à ouvrir (char * nom_f)
83   * varloc : pointeurs de parcours, les informations à copier
84   * sortie : code erreur fourni en parametre de sortie (int erreur)
85   */
86
87  int Lecture(semaine_t ** p_sem, char * nom_f)
88  {
89      /*variable locales*/
90      FILE *            f_lecture = fopen(nom_f, "r");
91      int               taille = TMAX + 10;
92      char *            ligne = (char *) calloc(1,sizeof(char)*taille);
93
94      semaine_t *       new_sem = NULL; /*pointeurs de parcours des listes*/
95      semaine_t **      prec_sem = NULL;
96      action_t *        new_act = NULL;
97      action_t **       prec_act = NULL;
98
99      int               erreur = 0;
100
```

```
    char              chaine_sem[7]; /*copies des informations*/
    char              chaine_act[TACT + 1];
    char              dechet[100];

    if (f_lecture && ligne)
    {
        /*initialisation*/
        prec_sem = p_sem;

        /*creation de la structure*/
        while ((fgets(ligne,taille,f_lecture) != NULL))
        {
            sscanf(ligne,"%6c%13c%100c",chaine_sem,chaine_act,dechet);

            /*initialisation des chaines*/
            chaine_sem[6] = '\0';
            chaine_act[strlen(chaine_act)] = '\0';

            if ((*p_sem == NULL) || (strcmp(new_sem->chaine_sem,chaine_sem) != 0))
            {
                /*insertion de semaine_t*/
                new_sem = InsererSem(prec_sem,chaine_sem);
                prec_sem = &((*new_sem).suiv);
                prec_act = &((*new_sem).action_s);

                /*insertion de action_t*/
                new_act = InsererAct(prec_act,chaine_act);
                prec_act = &((*new_act).suiv);
            }
            else
            {
                /*insertion de action_t*/
                new_act = InsererAct(prec_act,chaine_act);
                prec_act = &((*new_act).suiv);

            }
        }

        fclose(f_lecture);
        free(ligne);
    }
    else
    {
        printf("Ouverture du fichier de lecture impossible !\n");
        erreur = 1;
    }

    return erreur;
}
```

1.4.9 BILATÈRE : bilatere.c

```
/* Malrin Vincent
 * Laurent Valentin
 * 25/03/2013
 * BILATERE : bilatere.c
 */

/*INCLUDE*/
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "agenda.h"

/* FONCTION : Transformation
 * Transforme la liste des semaines en liste bilatère.
 *
 * entrée : le pointeur de debut de la liste chainée (semaine_t * p_sem)
 *          le pointeur de debut de la liste bilatère (semaine_t * p_bil)
 * varloc : le pointeurs precedents (semaine_t ** prec, bilatere_t ** prec_bil)
 *          le pointeur courant de la liste bilatère (bilatere_t * nouv)
 * sortie : le pointeur de debut de la liste bilatère (semaine_t * p_bil)
 */

bilatere_t * TransfoBilatere (semaine_t ** p_sem, bilatere_t * p_bil)
{
    /*variables locales*/
    semaine_t **      prec = p_sem;
    bilatere_t **     prec_bil = &p_bil;
    bilatere_t *      nouv = NULL;

    /*boucle*/
    while (*prec)
    {
        nouv = (bilatere_t *) calloc (1,sizeof(bilatere_t)); /*création de la cellule bilatère*/

        strcpy((*nouv).chaine_sem,(**prec).chaine_sem); /*copie des données*/
        (*nouv).action_s = (**prec).action_s;

        prec_bil = InsertionBil(prec_bil,nouv); /*insertion*/

        prec = &(**prec).suiv; /*avance les prec*/
    }
    return p_bil;
}

/* FONCTION : InsertionBil
 * Insère une celulle .
 *
 * entrée : l'adresse du pointeur où il faut inserer (bilatere_t ** b)
 *          le pointeur de la cellule à inserer (bilatere_t * elt)
 * sortie : l'adresse de l'élement inséré (semaine_t ** adr_bil)
 */

bilatere_t ** InsertionBil(bilatere_t ** b, bilatere_t * elt)
{
    /*insertion*/
    elt->suiv = (**b).suiv;
    elt->prec = *b;
    (*b)->suiv->prec = elt;
    (*b)->suiv = elt;

    return &(**b).suiv;
}

/* FONCTION : LiberationBil
 * Libere toute les données allouées dans la structure bilatere.
 *
```

```
 * entree : le pointeur de debut de la liste bilatere (bilatere_t * p_bil)
 * varloc : les pointeurs courants (action_t * cour_act et bilatere_t * cour_bil)
 *         les pointeurs temporaires (action_t * tmp_act et bilatere_t * tmp_bil)
 * sortie : rien
 */

void LiberationBil(bilatere_t * p_bil)
{
    /*variables locales*/
    bilatere_t *      cour_bil = (*p_bil).suiv;
    bilatere_t *      tmp_bil = NULL;
    int               verif = 0;

    /*parcours*/
    while ((cour_bil != NULL) && ((*cour_bil).action_s != NULL))
    {

        /* ATTENTION : les pointeurs (action_t *) ne sont pas libérés !
         * Ils seront libérés dans la fonction Liberation.
         */

        tmp_bil = cour_bil;

        /*on avance le pointeur de semaine_t*/
        cour_bil = (*cour_bil).suiv;

        /*libération de la structure semaine_t*/
        (*tmp_bil).action_s = NULL;
        (*tmp_bil).prec = NULL;
        (*tmp_bil).suiv = NULL;
        free(tmp_bil);

    }
    (*p_bil).suiv = NULL;
    (*p_bil).prec = NULL;
}
```

1.4.10 MAKEFILE

```
# compilateur
CC = gcc

# options de compilation
CFLAGS = -Wall -ansi -pedantic -Wextra -g -O2

# options de l'edition des liens
LDFLAGS = -g

# nom de l'executable a generer
EXEC = agenda

# liste des fichiers objets
OBJ = agenda.o afficher.o bilatere.o insertion.o recherche.o suppression.o ecriture.o motif.o

all : $(EXEC)

# regle de production finale
agenda : $(OBJ)
  $(CC) $(OBJ) -o $@ $(LDFLAGS)

# regle de production pour chaque fichier
agenda.o : agenda.c agenda.h
  $(CC) -c $< $(CFLAGS)

afficher.o : afficher.c agenda.h
  $(CC) -c $< $(CFLAGS)

bilatere.o : bilatere.c agenda.h
  $(CC) -c $< $(CFLAGS)

ecriture.o : ecriture.c agenda.h
  $(CC) -c $< $(CFLAGS)

insertion.o : insertion.c agenda.h
  $(CC) -c $< $(CFLAGS)

recherche.o : recherche.c agenda.h
  $(CC) -c $< $(CFLAGS)

suppression.o : suppression.c agenda.h
  $(CC) -c $< $(CFLAGS)

motif.o : motif.c agenda.h
  $(CC) -c $< $(CFLAGS)

# regle de suppression des fichiers
clean :
  rm $(OBJ)
```

1.5 Jeux de tests

1.5.1 Cas à tester

Lecture et Insertion (Question 1)

Cas à tester :

- Cas de fichier non valide (pour la lecture)
- Cas liste vide
- Cas cellule unique
- Cas général (10 cellules)

Recherche et Suppression (Question 3)

Cas à tester :

- Cas liste vide
- Cas cellule unique
- Cas général (10 cellules)
  - Recherche en début
  - Recherche au milieu
  - Recherche en fin
- Tests sur des dates non valides

Recherche de Motif (Question 2)

Cas à tester :

- Cas liste vide
- Cas général (10 cellules)
  - Recherche en début
  - Recherche au milieu
  - Recherche en fin
- Test sur un motif non valide
- Test sur un motif multiple

Ecriture (Question 2)

Cas à tester :

- Cas liste vide
- Cas général (10 cellules)

Liste Bilatère (Question 4)

Cas à tester :

- Cas liste vide
- Cas cellule unique
- Cas général (3 cellules, pour facilité l'affichage sur DDD)

1.5.2 Fichiers en entrée

Test de listes vides : donnee0.txt

Test de cellule unique : donnee1.txt

```
201305401fin de la premiere question
```

Test du cas général (10 cellules) : donnee10.txt

```
201342108Calcul
201342110Automates
201342113Proba
201343208TP de SDD
201343210Systemes
201344308TP de BDD
201344313Anglais
201345408Circuits
201403515TP de physique
201403517Liberte
```

Test du cas général (4 cellules) : donnee4.txt

```
201305401fin de la premiere question
201305402joue a la balancoire
201321110TP de SDD
201371110TP de BDD
```

Fichier d'écriture : donnee_e.txt

1.5.3 Execution

Lecture et Insertion (Question 1)

– Cas Fichier invalide : ./agenda 1 fichier_innexistant.txt

```
Ouverture du fichier de lecture impossible !
-------------------------------------------
Fin du programme
```

– Cas liste vide : ./agenda 1 donnee0.txt

```
-------------------------------------------
Fin du programme
```

– Cas liste unique : ./agenda 1 donnee1.txt

```
ANNEE : 2013
SEMAINE : 05
JOUR : 4
HEURE : 01
LIBELLE : fin de la

-------------------------------------------
Fin du programme
```

– Cas général : ./agenda 1 donnee10.txt

```
ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 08
LIBELLE : Calcul

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 10
LIBELLE : Automates

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 13
LIBELLE : Proba

ANNEE : 2013
SEMAINE : 43
JOUR : 2
HEURE : 08
LIBELLE : TP de SDD

ANNEE : 2013
```

```
SEMAINE : 43
JOUR : 2
HEURE : 10
LIBELLE : Systemes

ANNEE : 2013
SEMAINE : 44
JOUR : 3
HEURE : 08
LIBELLE : TP de BDD

ANNEE : 2013
SEMAINE : 44
JOUR : 3
HEURE : 13
LIBELLE : Anglais

ANNEE : 2013
SEMAINE : 45
JOUR : 4
HEURE : 08
LIBELLE : Circuits

ANNEE : 2014
SEMAINE : 03
JOUR : 5
HEURE : 15
LIBELLE : TP de phys

ANNEE : 2014
SEMAINE : 03
JOUR : 5
HEURE : 17
LIBELLE : Liberte

------------------------------------------
Fin du programme
```

Recherche (Question 3)

- Recherche -> Cas liste vide : ./agenda 2 donnee0.txt 201342108

```
Aucune page n'est trouvee
Fin du programme
```

- Recherche -> Cas cellule unique : ./agenda 2 donnee1.txt 201342108

```
LIBELLE RECHERCHEE : fin de la
Fin du programme
```

- Cas général (recherche au début) : ./agenda 2 donnee10.txt 201342108 (libelle : Calcul)

```
LIBELLE RECHERCHEE : Calcul
Fin du programme
```

- Cas général (recherche au milieu) : ./agenda 2 donnee10.txt 201343210 (libelle : Systeme)

```
LIBELLE RECHERCHEE : Systemes
Fin du programme
```

- Cas général (recherche au fin) : ./agenda 2 donnee10.txt 201403517 (libelle : Liberte)

```
LIBELLE RECHERCHEE : Liberte
Fin du programme
```

- Cas général (date incorrecte) : ./agenda 2 donnee10.txt 201343209

```
Aucune page n'est trouvee
Fin du programme
```

Suppression (Question 3)

- Suppression -> Cas liste vide : ./agenda 3 donnee0.txt 201342108

```
Aucune page n'est supprimee
Fin du programme
```

- Suppression -> Cas cellule unique : ./agenda 3 donnee1.txt 201342108

```
LIBELLE SUPPRIMEE : fin de la

-----------------------------------------
Fin du programme
```

- Cas général (Suppression au début) : ./agenda 3 donnee10.txt 201343108 (libelle : Calcul)

```
LIBELLE SUPPRIMEE : Calcul

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 10
LIBELLE : Automates

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 13
LIBELLE : Proba

ANNEE : 2013
SEMAINE : 43
JOUR : 2
HEURE : 08
LIBELLE : TP de SDD

ANNEE : 2013
SEMAINE : 43
JOUR : 2
HEURE : 10
LIBELLE : Systemes

ANNEE : 2013
SEMAINE : 44
JOUR : 3
HEURE : 08
LIBELLE : TP de BDD

ANNEE : 2013
SEMAINE : 44
JOUR : 3
HEURE : 13
LIBELLE : Anglais

ANNEE : 2013
```

```
SEMAINE : 45
JOUR : 4
HEURE : 08
LIBELLE : Circuits

ANNEE : 2014
SEMAINE : 03
JOUR : 5
HEURE : 15
LIBELLE : TP de phys

ANNEE : 2014
SEMAINE : 03
JOUR : 5
HEURE : 17
LIBELLE : Liberte

-------------------------------------------
Fin du programme
```

– Cas général (Suppression au milieu) : ./agenda 3 donnee10.txt 201343208 (libelle : TP de SDD)

```
LIBELLE SUPPRIMEE : TP de SDD

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 08
LIBELLE : Calcul

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 10
LIBELLE : Automates

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 13
LIBELLE : Proba

ANNEE : 2013
SEMAINE : 43
JOUR : 2
HEURE : 10
LIBELLE : Systemes

ANNEE : 2013
SEMAINE : 44
JOUR : 3
HEURE : 08
LIBELLE : TP de BDD

ANNEE : 2013
SEMAINE : 44
JOUR : 3
```

```
HEURE : 13
LIBELLE : Anglais

ANNEE : 2013
SEMAINE : 45
JOUR : 4
HEURE : 08
LIBELLE : Circuits

ANNEE : 2014
SEMAINE : 03
JOUR : 5
HEURE : 15
LIBELLE : TP de phys

ANNEE : 2014
SEMAINE : 03
JOUR : 5
HEURE : 17
LIBELLE : Liberte

-------------------------------------------
Fin du programme
```

– Cas général (Suppression au fin) : ./agenda 3 donnee10.txt 201403517 (libelle : Liberte)

```
LIBELLE SUPPRIMEE : Liberte

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 08
LIBELLE : Calcul

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 10
LIBELLE : Automates

ANNEE : 2013
SEMAINE : 42
JOUR : 1
HEURE : 13
LIBELLE : Proba

ANNEE : 2013
SEMAINE : 43
JOUR : 2
HEURE : 08
LIBELLE : TP de SDD

ANNEE : 2013
SEMAINE : 43
JOUR : 2
HEURE : 10
LIBELLE : Systemes
```

```
32
33 ANNEE : 2013
34 SEMAINE : 44
35 JOUR : 3
36 HEURE : 08
37 LIBELLE : TP de BDD
38
39 ANNEE : 2013
40 SEMAINE : 44
41 JOUR : 3
42 HEURE : 13
43 LIBELLE : Anglais
44
45 ANNEE : 2013
46 SEMAINE : 45
47 JOUR : 4
48 HEURE : 08
49 LIBELLE : Circuits
50
51 ANNEE : 2014
52 SEMAINE : 03
53 JOUR : 5
54 HEURE : 15
55 LIBELLE : TP de phys
56
57 ------------------------------------------
58 Fin du programme
```

– Cas général (date incorrecte) : ./agenda 3 donnee10.txt 201343209

```
1 Aucune page n'est supprimee
2 Fin du programme
```

Ecriture (Question 2)

– Cas liste vide : ./agenda 4 donnee0.txt donnee_e0.txt

```
Fin du programme
```

Affichage de fichier donnee_e0.txt

– Cas général : ./agenda 4 donnee10.txt donnee_e1.txt

```
Fin du programme
```

Affichage du fichier donnee_e1.txt :

```
201342108Calcul  -
201342110Automates -
201342113Proba   -
201343208TP de SDD -
201343210Systemes -
201344308TP de BDD -
201344313Anglais -
201345408Circuits -
201403515TP de phys-
201403517Liberte -
```

Bilatere (Question 4)

– Cas liste vide : ./agenda 5 donnee0.txt

```
Fin du programme
```

Après l'execution de la fonction TransfoBilatere on a sur DDD :

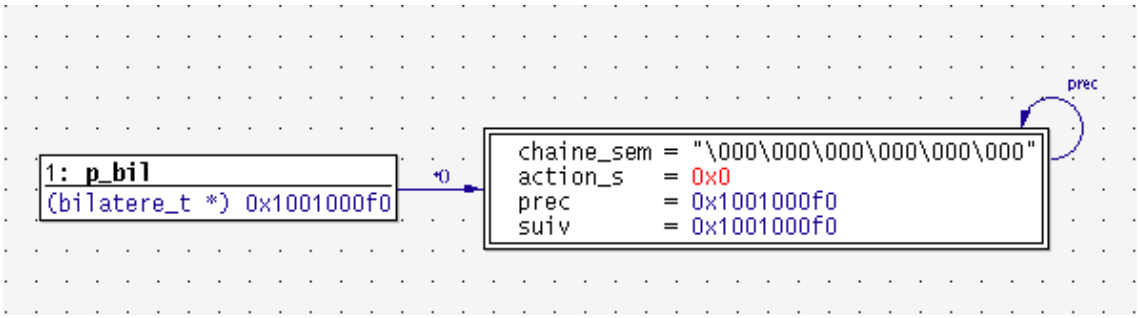


FIGURE 1.1 – Cas liste vide

– Cas cellule unique : ./agenda 5 donnee1.txt

```
Fin du programme
```

Après l'execution de la fonction TransfoBilatere on a sur DDD :

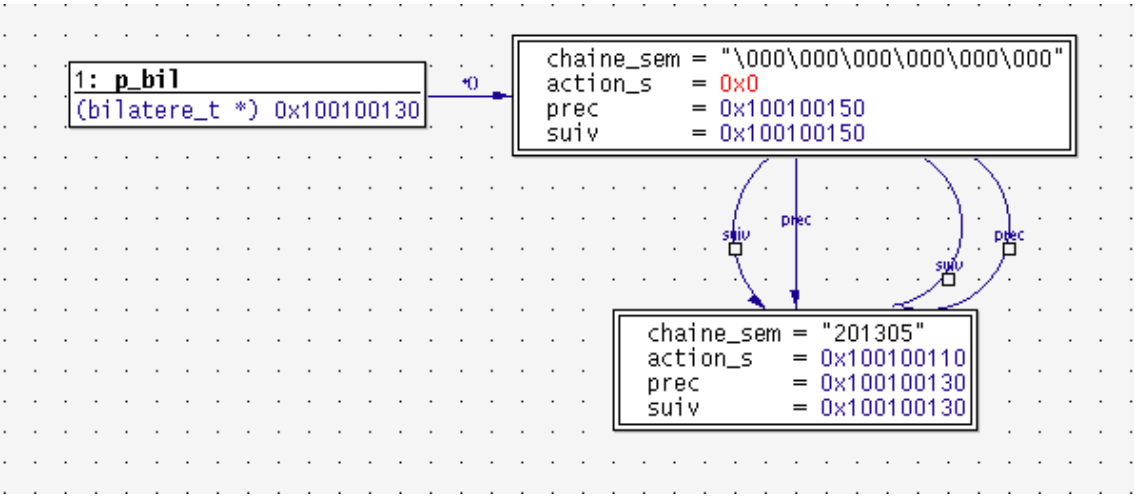


FIGURE 1.2 – Cas liste vide

Num	Expression	State	Scope	Address
1:	p_bil	enabled	TransfoBilatere	0x7fff5fbff748
2:	*p_bil	enabled	TransfoBilatere	0x100100130
3:	*p_bil->prec	enabled	TransfoBilatere	0x100100150
4:	*p_bil->suiv	alias of 3	TransfoBilatere	0x100100150
5:	*p_bil->prec->suiv	alias of 2	TransfoBilatere	0x100100130
6:	*p_bil->prec->prec	alias of 2	TransfoBilatere	0x100100130

– Cas général (4 cellules) : ./agenda 5 donnee4.txt

```
Fin du programme
```

Après l'execution de la fonction TransfoBilatere on a sur DDD :

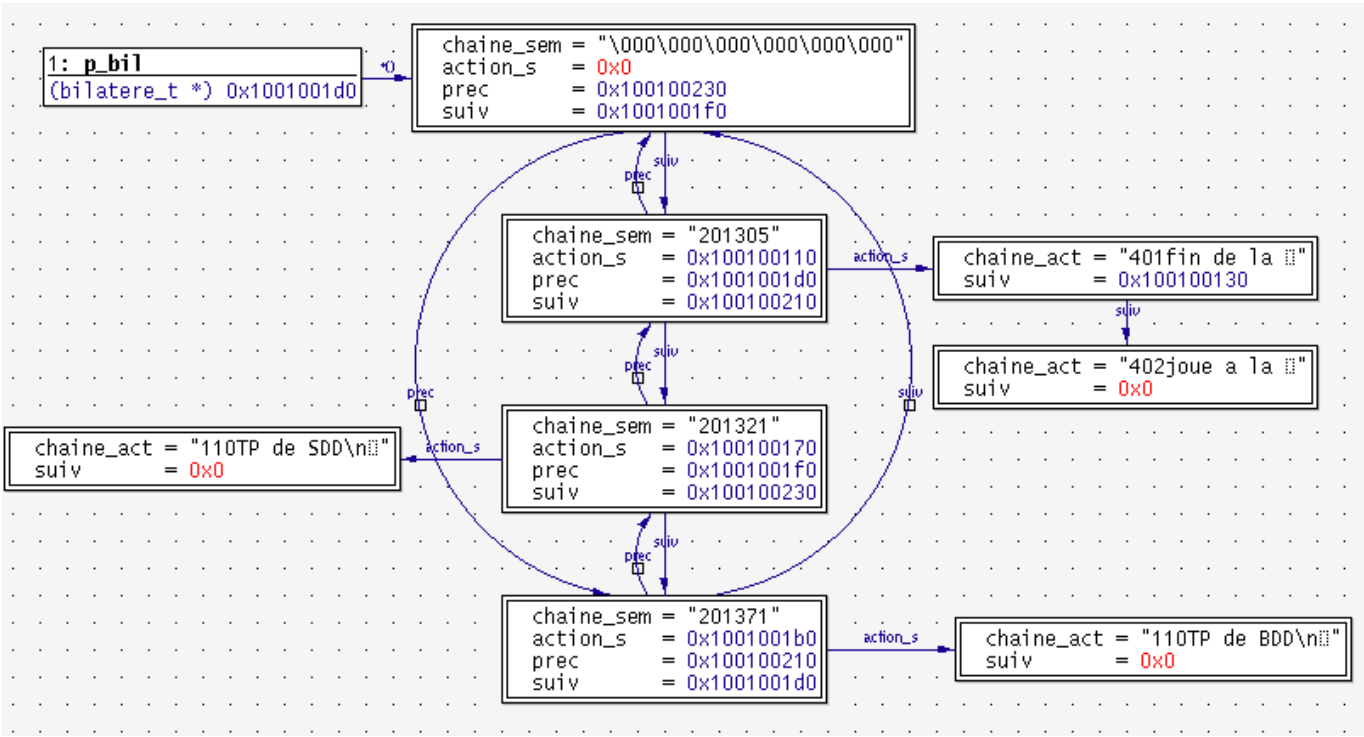


FIGURE 1.3 – Cas liste vide

Num	Expression	State	Scope	Address
1:	p_bil	enabled	TransfoBilatere	0x7fff5fbff728
2:	*p_bil	enabled	TransfoBilatere	0x1001001d0
8:	*p_bil->suiv->prec	alias of 2	TransfoBilatere	0x1001001d0
9:	*p_bil->suiv	enabled	TransfoBilatere	0x1001001f0
10:	*p_bil->suiv->suiv	enabled	TransfoBilatere	0x100100210
11:	*p_bil->suiv->suiv->suiv	enabled	TransfoBilatere	0x100100230
14:	*p_bil->suiv->prec	alias of 2	TransfoBilatere	0x1001001d0
17:	*p_bil->suiv->suiv->prec	alias of 9	TransfoBilatere	0x1001001f0
18:	*p_bil->suiv->suiv->suiv->prec	alias of 10	TransfoBilatere	0x100100210
19:	*p_bil->suiv->suiv->suiv->suiv	alias of 2	TransfoBilatere	0x1001001d0
20:	*p_bil->prec	alias of 11	TransfoBilatere	0x100100230

Recherche Motif (Question 2)

- Cas liste vide : ./agenda 6 donnee0.txt SDD

```
Aucune date correspondante
Fin du programme
```

- Cas général (recherche en début) : ./agenda 6 donnee10.txt Calcul

```
201342108
Fin du programme
```

- Cas général (recherche au milieu) : ./agenda 6 donnee10.txt SDD

```
201343208
Fin du programme
```

- Cas général (recherche en fin) : ./agenda 6 donnee10.txt Liberte

```
201403517
Fin du programme
```

- Cas général et libelle innexistant : ./agenda 6 donnee10.txt truc

```
Aucune date correspondante
Fin du programme
```

- Cas général et libelle multiple : ./agenda 6 donnee10.txt TP

```
201343208
201344308
201403515
Fin du programme
```