ISIMA Première Année

Compte-Rendu de TP

Structures de Données

Gestion de news à partir d'une liste chaînée

Benjamin BARBESANGE
Pierre-Loup PISSAVY
Groupe G21

Enseignant :
Michelle CHABROL

février 2015

ISIMA

1 | Présentation

Le but de ce TP est de concevoir un ensemble de fonctions permettant de gérer des news sous forme de messages, chacun d'entre eux ayant une date de début et de fin de validité. On doit faire usage d'une liste chaînée. Les news sont ordonnées dans la liste chaînée selon l'ordre décroissant de la date de début (de la plus récente à la plus ancienne).

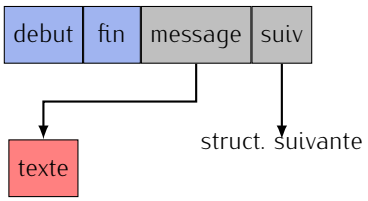
Les messages et informations satellites sont enregistrés dans un fichier, à raison d'une ligne par message. Ce fichier est supposé correct.

Les opérations suivantes sont permises :

- Charger une liste depuis un fichier,
- Sauvegarder une liste dans un fichier,
- Afficher les messages du jour,
- Supprimer les messages obsolètes,
- Modifier une date de début sur tous les messages,
- Afficher tous les messages contenant une chaîne particulière.

1.1 Structure de données employée

(a) Structure utilisée



(b) Code

Code C

```c
typedef struct _news_t {
  int  debut;
  int  fin;
  char *texte;
  struct _news_t *suiv;
} news_t;
```

FIGURE 1.1 – Structure et code correspondant

1.2 Organisation du code source

Nous avons défini deux modules, le premier est dédié à la gestion des listes chaînées (adjonction, recherche, suppression etc.), et le second à la gestion des news (lecture, sauvegarde, modifications etc.). Enfin, le programme principal utilise conjointement ces deux modules pour réaliser le traitement voulu.

1.2.1 Gestion de liste chaînée

- `src/liste_news.h`
- `src/liste_news.c`

1.2.2 Gestion de news

- `src/gestion_news.h`
- `src/gestion_news.c`

1.2.3 Programme principal

- `src/main.c`

2 | Détails du programme

2.1 Gestion de liste chaînée

Code C

```c
/*  liste_news.h
  Header

  -----| GESTION DE NEWS PAR LISTE CHAINEE |-----

  BARBESANGE Benjamin,
  PISSAVY Pierre-Loup

  ISIMA 1ere Annee, 2014-2015
*/

#ifndef __LISTE_NEWS_H__
#define __LISTE_NEWS_H__

  #include <string.h>

  typedef struct _news_t {
    int  debut;
    int  fin;
    char *texte;
    struct _news_t *suiv;
  } news_t;

  typedef struct _news_t cell_t;

  cell_t ** rech_prec(cell_t **, int, short int*);
  void supp_cell(cell_t **);
  void liberer_liste(cell_t **);
  void ins_cell(cell_t **, cell_t *);
  cell_t * creer_cell(int, int, char *);

#endif
```

Code C

```c
/*  liste_news.c
  Fonctions de gestion de la liste chainee

  -----| GESTION DE NEWS PAR LISTE CHAINEE |-----

  BARBESANGE Benjamin,
  PISSAVY Pierre-Loup

  ISIMA 1ere Annee, 2014-2015
*/

#include <stdio.h>
#include <stdlib.h>
#include "liste_news.h"

/*  void adj_cell(cell_t **prec, cell_t *elt)
  Ajoute une cellule apres un element partir d'un pointeur sur l'element
  et d'un pointeur sur le pointeur de l'element apres lequel ajouter

  Entrees :
    cell_t **prec : pointeur sur le pointeur de l'element apres lequel ajouter
    cell_t *elt : pointeur sur l'element a ajouter a la liste chainee

  Sortie :
    Aucune
*/
void adj_cell(cell_t **prec, cell_t *elt) {
  elt->suiv = (*prec);
  (*prec) = elt;
}

/*  cell_t ** rech_prec(cell_t **liste, int debut, short int *existe)
  Recherche le precedent d'un element dans la liste chainee a partir de la
  date de debut de message

  Entrees :
    cell_t **liste : pointeur sur le pointeur du premier element de la liste chainee
    int debut : date de debut de message que l'on souhaite inserer, sous la forme AAAAMMJJ
          on cherchera donc les messages dont la date est immediatement superieure
    short int *existe : variable en entre/sortie indiquant si on a ou pas de message ayant la date de debut
      0 : il n'y a pas de message avec cette date de debut
      1 : il y a au moins un message

  Sortie :
    cell_t ** : pointeur sur le pointeur de l'element precedent
*/
cell_t ** rech_prec(cell_t **liste, int debut, short int *existe) {
  cell_t **prec = liste;
  while ((*prec) && (*prec)->debut > debut) {
    prec = &((*prec)->suiv);
  }
  /* Booleen de presence     */
  /* 1 : present             */
  /* 0 : absent              */
```

```c
  *existe = (*prec && (*prec)->debut == debut)?1:0;
  return prec;
}

/*  void supp_cell(cell_t **prec)
  Permet de supprimer un element dans la liste chainee a partir
  de son precedent

  Entrees :
    cell_t **prec : pointeur sur le pointeur de l'element precedent l'element a supprimer

  Sortie :
    Aucune
*/
void supp_cell(cell_t **prec) {
  cell_t *elt = *prec;
  *prec = elt->suiv;
  free(elt->texte);
  free(elt);
}

/*  void liberer_liste(cell_t **liste)
  Libere les allocations memoires de la liste

  Entrees :
    cell_t **liste : pointeur sur le pointeur du premier element de la liste chainee

  Sortie :
    Aucune
*/
void liberer_liste(cell_t **liste) {
  while (*liste) {
    supp_cell(liste);
  }
  *liste = NULL;
}

/*  void ins_cell(cell_t **liste, cell_t *elt)
  Permet d'inserer une cellule a la bonne place dans la liste chainee
  Les messages sont tries par ordre decroissant des date de debut

  Entrees :
    cell_t **liste : pointeur sur le pointeur du premier element de la liste chainee
    cell_t *elt : pointeur sur l'element a inserer dans la liste chainee

  Sortie :
    Aucune
*/
void ins_cell(cell_t **liste, cell_t *elt) {
  short int existe;
  cell_t **prec = rech_prec(liste,elt->debut,&existe);
  adj_cell(prec,elt);
}

/*  news_t * creer_cell(int debut, int fin, char *message)
  Permet de creer un element de la liste chainee a partir de
```

```c
  la date de debut et de fin de validite ainsi que le texte du message

  Entrees :
    int debut : date de debut de validite de message sous la forme AAAAMMJJ
    int fin : date de fin de validite de message sous la forme AAAAMMJJ
    char *message : texte du message

  Sortie :
    news_t* : pointeur sur l'element cree
*/
news_t * creer_cell(int debut, int fin, char *message) {
  news_t *elt = (news_t*) malloc(sizeof(news_t));
  if (elt) {
    elt->debut = debut;
    elt->fin = fin;
    elt->texte = (char*) malloc((strlen(message)+1)*sizeof(char));
    strcpy(elt->texte,message);
  }
  return elt;
}
```

2.2 Gestion de news

Code C

```c
/*  gestion_news.h
  Header

  -----| GESTION DE NEWS PAR LISTE CHAINEE |-----

  BARBESANGE Benjamin,
  PISSAVY Pierre-Loup

  ISIMA 1ere Annee, 2014-2015
*/

#ifndef __GESTION_NEWS_H__
#define __GESTION_NEWS_H__

  #include "liste_news.h"

  int charger(cell_t **, char *);
  int sauver(cell_t *, char *);
  int getDate();
  void afficher_messages_date(cell_t *, int);
  void afficher_message(cell_t *);
  void afficher_liste(cell_t *);
  void afficher_messages_jour(cell_t *);
  void afficher_messages_motif(cell_t *, char *);
  void supprimer_obsoletes(cell_t **);
  void remplacer_date(cell_t **, int, int);

#endif
```

Code C

```c
/*  gestion_news.c
  Fonctions de gestion des news

  -----| GESTION DE NEWS PAR LISTE CHAINEE |-----

  BARBESANGE Benjamin,
  PISSAVY Pierre-Loup

  ISIMA 1ere Annee, 2014-2015
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include "gestion_news.h"

/* taille maximale de la chaine de caractere du message */
#define SIZE_BUF 118

char buf[SIZE_BUF+1];

/*  int charger(cell_t **liste, char *nom_fichier)
    Permet de charger les donnees d'un fichier passe en parametre
    dans une liste dont l'adresse du pointeur de tete est
    egalement passe en parametre

  Entrees :
    cell_t **liste : adresse du pointeur de tete de liste dans
    char *nom_fichier : chemin et nom du fichier a partir duquel charger la liste

  Sortie :
    Entier indiquant un code d'erreur
      0 si aucune erreur
      1 si le fichier ne s'est pas charge
      2 si probleme d'allocation d'un element dans la liste
*/
int charger(cell_t **liste, char *nom_fichier) {
  int ret = 1;
  FILE *fichier = fopen(nom_fichier,"r");
  int debut, fin;
  char *scan;
  cell_t *tmp;
  if (fichier) {
    ret = 0;
    while (!feof(fichier) && ret == 0 && fgets(buf,SIZE_BUF+1,fichier)) {
      /* suppression du caractere \n residuel si le texte fait moins de 100 caracteres */
      scan = strchr(buf,'\n');
      if (scan) {
        *scan = '\0';
      }
      sscanf(buf,"%d %d",&debut, &fin);
      tmp = creer_cell(debut,fin,&buf[18]);
      if (tmp) {
        ins_cell(liste,tmp);
```

```
      } else {
        /* Code erreur: allocation de cellule */
        ret = 2;
      }
    }
    fclose(fichier);
  }
  return ret;
}

/*   int sauver(cell_t *liste, char *nom_fichier)
  Sauvegarde les donnees de la liste chainee dans un fichier passe
  en parametre

  Entrees :
    cell_t *liste : pointeur sur le premier element de la liste chainee
    char *nom_fichier : chemin et nom du fichier dans lequel sauvegarder
              ce fichier sera cree et effacera tout autre fichier
              du meme nom dans le repertoire specifie s'il y en a

  Sortie :
    Entier renvoyant un code d'erreur
      0 si aucune erreur
      1 si probleme de creation du fichier
*/
int sauver(cell_t *liste, char *nom_fichier) {
  int ret = 1;
  cell_t *cour = liste;
  FILE *fichier = fopen(nom_fichier,"w+");
  if (fichier) {
    while (cour) {
      fprintf(fichier,"%d %d %s\n",cour->debut,cour->fin,cour->texte);
      cour = cour->suiv;
    }
    fclose(fichier);
    ret = 0;
  }
  return ret;
}

/*  int getDate()
  Retourne la date du jour au format AAAAMMJJ

  Entrees :
    Aucune

  Sortie :
    Entier representant la date du jour de la forme AAAAMMJJ
*/
int getDate() {
  time_t now = time(NULL);
  int datejour;
  struct tm t = *localtime(&now);
  datejour = (t.tm_year+1900)*10000+(t.tm_mon+1)*100+t.tm_mday;
  return datejour;
}
```

```c

/*  void afficher_message(cell_t *m)
  Affiche le message d'un element de la liste chainee sous la forme
  "Debut : *debut*, Fin: *fin*, Message: *texte*"

  Entrees :
    cel_t *m : pointeur sur un element de la liste chainee

  Sortie :
    Aucune
*/
void afficher_message(cell_t *m) {
  printf("Debut: %d, Fin: %d, Message: %s\n",m->debut,m->fin,m->texte);
}

/*  void afficher_liste(cell_t *l)
  Affiche le contenu integral de la liste suivant le schema de la fonction
  afficher_message(cell_t *m)

  Entrees :
    cell_t *l : pointeur sur le premier element de la liste

  Sortie :
    Aucune
*/
void afficher_liste(cell_t *l) {
  cell_t *cour = l;
  while (cour) {
    afficher_message(cour);
    cour = cour->suiv;
  }
}

/*  void afficher_messages_date(cell_t *liste, int date)
  Affiche les messages de la liste correspondant a la date passee en parametre

  Entrees :
    cell_t *liste : pointeur sur le premier element de la liste chainee
    int date : date de debut des elements a traiter, de la forme AAAAMMJJ

  Sortie :
    Aucune
*/
void afficher_messages_date(cell_t *liste, int date) {
  cell_t *cour = liste;
  while (cour && cour->debut > date) {
    cour = cour->suiv;
  }
  while (cour && cour->debut == date) {
    afficher_message(cour);
    cour = cour->suiv;
  }
  while (cour) {
    while (cour && cour->fin >= date) {
      afficher_message(cour);
      cour = cour->suiv;
```

```
    }
    if (cour) {
      cour = cour->suiv;
    }
  }
}

/*  void afficher_messages_jour(cell_t *liste)
  Affiche les messages de la liste correspondant a la date du jour

  Entrees :
    cell_t *liste : pointeur sur le premier element de la liste chainee

  Sortie :
    Aucune
*/
void afficher_messages_jour(cell_t *liste) {
  afficher_messages_date(liste,getDate());
}

/*  void afficher_messages_motif(cell_t *liste, char *motif)
  Affiche les messages de la liste chainee correspondant a un motif
  passe en parametre

  Entrees :
    cell_t *liste : pointeur sur le premier element de la liste chainee
    char *motif : chaine de caractere representant le motif a chercher dans
            les messages de la liste

  Sortie :
    Aucune
*/
void afficher_messages_motif(cell_t *liste, char *motif) {
  cell_t *cour = liste;
  while (cour) {
    if (strstr(cour->texte,motif)) {
      afficher_message(cour);
    }
    cour = cour->suiv;
  }
}

/*  void supprimer_obsoletes(cell_t **liste)
  Supprime les messages devenus obsoletes dans la liste chainee
  Les messages sont obsoletes si leur date de fin est anterieure a la date du jour

  Entrees :
    cell_t **liste : adresse du pointeur sur le premier element de la liste chainee

  Sortie :
    Aucune
*/
void supprimer_obsoletes(cell_t **liste) {
  int date = getDate();
  cell_t **prec = liste;
  while (*prec != NULL) {
```

```
    if ((*prec)->fin < date) {
      supp_cell(prec);
    } else {
      prec = &((*prec)->suiv);
    }
  }
}

/*  void remplacer_date(cell_t **liste, int date, int nvdate)
  Remplace la date de debut des messages ayant une date passee en parametre par une
  autre date aussi passee en parametre
  Cette fonction s'assure egalement que la liste reste triee

  Entrees :
    cell_t **liste : adresse du pointeur sur le premier element de la liste chainee
    int date : date de debut des elements a traiter, de la forme AAAAMMJJ
    int nvdate : nouvelle date de debut a assigner aux elements

  Sortie :
    Aucune
*/
void remplacer_date(cell_t **liste, int date, int nvdate) {
  cell_t *cour = *liste;
  while (cour && cour->debut > date) {
    cour = cour->suiv;
  }
  while (cour && cour->debut == date) {
    if (cour->fin >= nvdate) {
      cour->debut = nvdate;
    }
    cour = cour->suiv;
  }
  sauver(*liste,"/tmp/tmp_tp1_sdd.list");
  liberer_liste(liste);
  charger(liste,"/tmp/tmp_tp1_sdd.list");
}
```

2.3 Programme principal

Code C

```c
/*  main.c
  Fichier principal permettant les tests

  -----| GESTION DE NEWS PAR LISTE CHAINEE |-----

  BARBESANGE Benjamin,
  PISSAVY Pierre-Loup

  ISIMA 1ere Annee, 2014-2015
*/

#include <stdio.h>
#include "gestion_news.h"

int main(int argc, char *argv[]) {
  int old_deb = 20150215;
  int new_deb = 20150326;
  cell_t *liste = NULL;
  if (argc > 1) {
    charger(&liste,argv[1]);
  }
  if (liste) {
    printf("Affichage de la liste apres recuperation\n");
    afficher_liste(liste);

    printf("\nAffichage des messages du jour\n");
    afficher_messages_jour(liste);

    printf("\nSuppression des messages obsoletes\n");
    supprimer_obsoletes(&liste);
    afficher_liste(liste);

    printf("\nModification des dates de debut: %d -> %d\n",old_deb,new_deb);
    remplacer_date(&liste,old_deb,new_deb);
    afficher_liste(liste);

    liberer_liste(&liste);
  }
  return 0;
}
```

3 | Principes et lexiques des fonctions

3.1 Gestion des news

La gestion des news s'effectue grace aux fichiers `gestion_news.c` et `gestion_news.h`.

3.1.1 charger

```
Algorithme Charger (Principe)
Début
    On ouvre en lecture seule le fichier dont le nom est passé en paramètre;
    On initialise le code de retour à 1;   [ erreur d'ouverture de fichier ]
    Si on a pu l'ouvrir Alors
        Le code de retour passe à 0;   [ pas de problème d'ouverture de fichier ]
        TantQue l'on n'arrive pas à la fin du fichier Faire
            On lit une ligne;
            On stocke les date de début et de fin de message dans des variables;
            On stocke le message dans une chaîne de caractère;
            On crée une nouvelle cellule à partir des variables ci-dessus récupérées;
            Si on a pu créer la cellule Alors
                On l'insère dans la liste chaînée;
            Sinon
                Le code de retour passe a 2;   [ erreur allocation de cellule ]
            FinSi;
        FinTantQue;
        On ferme le fichier;
    FinSi;
    Retourner le code d'erreur;
Fin
```

Lexique :

ret : *entier qui retourne un code d'erreur*
- *0 si aucune erreur*
- *1 si problème lors de l'ouverture du fichier*
- *2 si problème lors de la création de la cellule*

*fichier : *descripteur de fichier, ouvert a partir du nom passé en paramètres*
debut, fin : *entiers servant à stocker les dates de début et de fin de message lors du traitement du fichier*
*scan : *chaîne de caractère permettant de stocker le message lors du traitement du fichier*
*tmp : *pointeur de cellule qui servira de cellule temporaire à chaîner dans la liste*

3.1.2 sauver

Algorithme Sauver (Principe)
Début
On crée un nouveau fichier à partir du nom donné en paramètre;
On initialise le code de retour à 1; *[erreur de création de fichier]*
Si l'on a pu créer le fichier Alors
TantQue l'on n'est pas à la fin de la liste Faire
On écrit les élément de la cellule en cours de traitement dans le fichier;
On passe à la cellule suivante;
FinTantQue;
On ferme le fichier;
On passe le code de retour à 0; *[pas d'erreur]*
FinSi;
Retourner le code d'erreur;
Fin

Lexique :

ret : *entier représentant le code d'erreur de la fonction*
- *0 aucune erreur*
- *1 erreur de création ou d'ouverture du fichier*

*cour : *pointeur sur la cellule en cours de traitement dans la liste chaînée*
*fichier : *descripteur de fichier crée à partir du nom passé en paramètre*

3.1.3 getDate

Cette fonction est fournie pour le TP, nous ne la détaillerons donc pas.

3.1.4 afficher_message

Algorithme afficher_message (Principe)
Début
    On écrit les données d'une cellule en récupérant ses attributs;
Fin

3.1.5 afficher_liste

Algorithme afficher_liste (Principe)
Début
    On initialise l'élément courant au début de la liste;
    TantQue l'élément courant n'est pas NULL Faire
        On affiche les élément de la cellule avec *afficher_message*;
        On passe à la cellule suivante;
    FinTantQue;
Fin

Lexique :
    *cour : *pointeur sur la cellule en cours de traitement dans la liste chaînée*

3.1.6 afficher_messages_date

Algorithme afficher_messages_date (Principe)
Début
    On initialise l'élément courant au début de la liste;
    TantQue la date de début de l'élément courant est supérieure à la date à traiter Faire
        On passe à la cellule suivante;
    FinTantQue;
    TantQue la date de début de l'élément courant est égale à la date recherchée Faire
        On affiche l'élément courant;
        On passe à la cellule suivante;
    FinTantQue;
    TantQue la liste n'est pas finie Faire
        TantQue la date de fin de l'élément courant est ultérieure à la date recherchée Faire
            On affiche l'élément courant;
            On passe à la cellule suivante;
        FinTantQue;

```
    Si la liste n'est pas finie Alors
        On passe à la cellule suivante;
    FinSi;
  FinTantQue;
Fin
```

3.1.7 afficher_messages_jour

Algorithme afficher_messages_date (Principe)
Début
  On appelle la fonction *afficher_messages_date* avec en paramètre la date du jour obtenue avec *getDate*;
Fin

3.1.8 afficher_messages_motif

Principe :

Lexique

-

3.1.9 supprimer_obsoletes

Principe :

Lexique

-

3.1.10 remplacer_date

Principe :

Lexique

-

3.2 Gestion de la liste chaînée

4 | Compte rendu d'exécution

4.1 Makefile

Makefile

```
#Compilateur et options de compilation
CC=gcc
CFLAGS=-Wall -ansi -pedantic -Wextra -g

#Fichiers du projet
SOURCES=main.c gestion_news.c liste_news.c
OBJECTS=$(SOURCES:.c=.o)

#Nom du programme
EXEC=programme

all: $(OBJECTS)
	$(CC) $(CFLAGS) $^ -o $(EXEC)

.c.o:
	$(CC) -c $(CFLAGS) $*.c
clean:
	rm $(OBJECTS) $(EXEC)
```

4.2 Jeux de tests

Exécution du programme avec le fichier suivant :

Test

```
20150215 20150416 Ajout liste vide
20150220 20150227 Ajout en tete
20150217 20150228 Ajout en milieu de liste
20150212 20150222 Ajout apres le dernier element
20150215 20150316 Evenement lambda
```

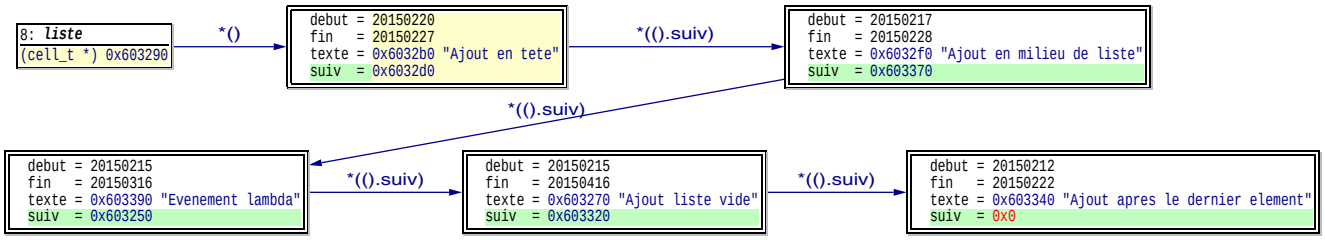


FIGURE 4.1 – Après lecture du fichier, contenu de `liste`

FIGURE 4.2 – Après suppression des messages obsolètes, contenu de `liste`

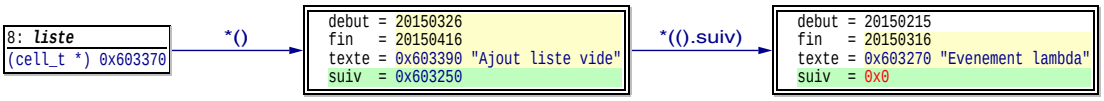


FIGURE 4.3 – Après modification de date de début, contenu de `liste`

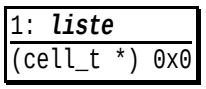


FIGURE 4.4 – Après suppression de la liste, contenu de `liste`

Terminal

```
Affichage de la liste apres recuperation
Debut: 20150220, Fin: 20150227, Message: Ajout en tete
Debut: 20150217, Fin: 20150228, Message: Ajout en milieu de liste
Debut: 20150215, Fin: 20150316, Message: Evenement lambda
Debut: 20150215, Fin: 20150416, Message: Ajout liste vide
Debut: 20150212, Fin: 20150222, Message: Ajout apres le dernier element

Affichage des messages du jour
Debut: 20150215, Fin: 20150316, Message: Evenement lambda
Debut: 20150215, Fin: 20150416, Message: Ajout liste vide

Suppression des messages obsoletes
Debut: 20150215, Fin: 20150316, Message: Evenement lambda
Debut: 20150215, Fin: 20150416, Message: Ajout liste vide

Modification des dates de debut: 20150215 -> 20150326
Debut: 20150326, Fin: 20150416, Message: Ajout liste vide
Debut: 20150215, Fin: 20150316, Message: Evenement lambda
```

FIGURE 4.5 – Exécution du programme sur la sortie standard

Terminal

```
==8377== Memcheck, a memory error detector
==8377== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==8377== Using Valgrind-3.10.0.SVN and LibVEX; rerun with -h for copyright info
==8377== Command: ./programme test_insert
==8377==
Affichage de la liste apres recuperation
Debut: 20150220, Fin: 20150227, Message: Ajout en tete
Debut: 20150217, Fin: 20150228, Message: Ajout en milieu de liste
Debut: 20150215, Fin: 20150316, Message: Evenement lambda
Debut: 20150215, Fin: 20150416, Message: Ajout liste vide
Debut: 20150212, Fin: 20150222, Message: Ajout apres le dernier element

Affichage des messages du jour
Debut: 20150215, Fin: 20150316, Message: Evenement lambda
Debut: 20150215, Fin: 20150416, Message: Ajout liste vide

Suppression des messages obsoletes
Debut: 20150215, Fin: 20150316, Message: Evenement lambda
Debut: 20150215, Fin: 20150416, Message: Ajout liste vide

Modification des dates de debut: 20150215 -> 20150326
Debut: 20150326, Fin: 20150416, Message: Ajout liste vide
Debut: 20150215, Fin: 20150316, Message: Evenement lambda
==8377==
==8377== HEAP SUMMARY:
==8377==     in use at exit: 0 bytes in 0 blocks
==8377==   total heap usage: 28 allocs, 28 frees, 4,674 bytes allocated
==8377==
==8377== All heap blocks were freed -- no leaks are possible
==8377==
==8377== For counts of detected and suppressed errors, rerun with: -v
==8377== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

FIGURE 4.6 – Exécution avec valgrind