ISIMA 1ère ANNEE

Rapport
TP1 de Structure de Données

Anaïs Darricarrere
Nada Boutadghart

13 mars 2020

Table des matières

Partie 1

Présentation générale

1.1 Description de l'objet du TP

Le but de ce TP est de gérer des messages à partir d'une liste chaînée. Cette liste chaînée doit contenir les différents messages obtenus à partir d'un fichier d'entrée et doivent être triés sur leurs dates de début de validité.

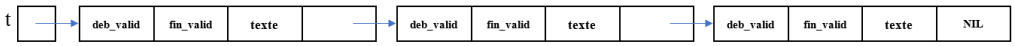
1.2 Description des structures

1.2.1 Structure construite : la liste chaînée

La liste chaînée est construite de la manière suivante :

deb_valid : Date de debut de validité du message (sous la forme d'un entier aaaammjj)
fin_valid : Date de fin de validité du message (de la même forme que deb_valid)
texte : Texte du message (maximum 100 caractères)
suivant : Pointeur vers le message suivant

Schéma de la structure



1.2.2 Fichier de données

Le fichier de données d'entrée est constitué de plusieurs lignes, et chacune d'entre elles est construite de la manière suivante :

deb_valid fin_valid texte

Les 3 éléments **deb_valid**, **fin_valid** et **texte** sont définit comme dans la liste chaînée.

Exemple d'une ligne du fichier : Avec le message "Il fait beau" qui est valide du 15 juin 2018 au 18 septembre 2065.

20180615 20650918 Il fait beau

1.3 Organisation du code source

1.3.1 Les fichiers d'entête

Lch.h contient :

- Des directives de préprocesseur permettant d'inclure les bibliothèques <stdio.h> et <stdlib.h> ;

- La déclaration du type cellule_t (pour une cellule de liste chaînée) ;
- Les prototypes des fonctions de gestion de liste chaînée ;

Code source :

```
/*------------------------------------------------------------------------------------------------------*/
/*                                    TP1 - Gestion de messages                                         */
/*                                                                                                      */
/*                                    Fichier d'entête Lch.h                                            */
/*                                                                                                      */
/* Déclaration des structures et prototypes des fonctions de gestion de listes chaînées                 */
/*------------------------------------------------------------------------------------------------------*/

#include <stdlib.h>
#include <string.h>

/*------------------------------------------------------------------------------------------------------*/
/* Déclaration de la structure cellule_t : bloc de la liste chaînée contenant :                         */
/*      deb_valid       date de début de validité du message (entier aaaammjj)                          */
/*      fin_valid       date de fin de validité du message (entier aaaammjj)                            */
/*      texte           texte du message (chaîne de caractères)                                         */
/*      suivant         pointeur vers le bloc suivant                                                   */
/*------------------------------------------------------------------------------------------------------*/

typedef struct cellule
{
    int deb_valid, fin_valid;
    char * texte;
    struct cellule * suivant;
}cellule_t;

/*------------------------------------------------------------------------------------------------------*/
/* Prototypes des fonctions                                                                             */
/*------------------------------------------------------------------------------------------------------*/

cellule_t * CREER_CELL(int deb, int fin, char * texte);

cellule_t ** RECH_PREC(int date_deb, cellule_t ** t);

void ADJ_CELL(cellule_t ** prec, cellule_t * nouv);

void INSERT_CELL(cellule_t * nouv, cellule_t ** t);

void SUPP_CELL(cellule_t ** prec);

void SUPP_LCH(cellule_t **t);
```

TP1.h contient :

- Des directives de préprocesseur permettant d'inclure les bibliothèques <stdio.h>, <stdlib.h>, <string.h> et <time.h>, ainsi que le fichier d'entête "Lch.h" ;
- Les prototypes des fonctions de gestion de messages ;

Code source :

```c
/*-------------------------------------------------------------------------*/
/*                            TP1 - Gestion de messages                    */
/*                            Fichier d'entête TP1.h                       */
/*                                                                         */
/* Déclaration des fonctions de gestion de messages                        */
/*-------------------------------------------------------------------------*/

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

/*-------------------------------------------------------------------------*/
/* Prototypes des fonctions                                                */
/*-------------------------------------------------------------------------*/

int LECTURE(char * nom_fichier, cellule_t ** t);

int SAUVEGARDE(cellule_t * t, char * nom_fichier);

int DATE_DU_JOUR();

void AFF_NON_EXP(cellule_t ** t);

void SUPP_MESS_EXP(cellule_t ** t);

void MODIF_DATE_DEB(cellule_t * t, int deb_init, int deb_nouv);
```

1.3.2 Les modules

Lch.c contient :

- Une directive de préprocesseur permettant d'inclure le fichier d'entête "Lch.h" ;
- Les codes des fonctions de gestion de listes chaînées déclarés dans le fichier Lch.h ;

Code source : Cf. 2 Détails de chaque fonction

TP1.c contient :

- Une directive de préprocesseur permettant d'inclure les fichiers d'entête "Lch.h" et "TP1.h" ;
- Les codes des fonctions de gestion de messages déclarés dans le fichier TP1.h ;

Code source : Cf. 2 Détails de chaque fonction

Main.c contient :

- Une directive de préprocesseur permettant d'inclure le fichier d'entête "TP1.h" ;
- Le code du programme principal qui teste les fonctions de gestions de messagerie ;

Code source : Cf. 2.3 Programme principal (Main.c)

Partie 2

Détails de chaque fonction

2.1 Programmes de gestion d'une liste chaînée

```
/*-------------------------------------------------------------------------------------------*/
/* CREER_CELL              Création d'une cellule de type cellule_t                          */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                          - Allouer un bloc de type cellule_t                              */
/*                          - Si l'allocation est réussie :                                  */
/*                              - Copier les valeurs entrées en paramètre dans la            */
/*                                nouvelle cellule                                           */
/*                          FIN                                                              */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : deb, fin    Deux entiers représentant respectivement les dates de            */
/*                          début et fin de validité du message                              */
/*              texte       Chaîne de caractères contenant le message                        */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              lg_texte    Longueur du message                                              */
/*                                                                                           */
/*  En sortie : nouv        Pointeur sur la cellule créée                                    */
/*-------------------------------------------------------------------------------------------*/

cellule_t * CREER_CELL(int deb, int fin, char * texte)
{
    cellule_t * nouv = NULL;
    int lg_texte;
    nouv = (cellule_t*)malloc(sizeof(cellule_t));
    if (nouv != NULL)
    {
        nouv->deb_valid = deb;
        nouv->fin_valid = fin;
        lg_texte = strlen(texte);
        nouv->texte = (char*)malloc(lg_texte*sizeof(char));
        strcpy(nouv->texte, texte);
    }
    return nouv;
}
```

```
/*-------------------------------------------------------------------------------------------*/
/* RECH_PREC                Recherche du précédent dans une liste chaînée triée              */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                          - Si la liste est non vide et la date de début de                */
/*                          validité du message est inférieure à la date recherchée :        */
/*                              - Avancer le pointeur du précedent et le pointeur            */
/*                                courant au bloc suivant dans la liste chaînée              */
/*                          FIN                                                              */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : date_deb    Entier représentant la date de début de la cellule à insérer     */
/*              t           Pointeur sur le pointeur de tête de la liste                     */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              cour        Pointeur sur la cellule courante                                 */
/*                                                                                           */
/*  En sortie : prec        Pointeur sur le précédent de la cellule recherchée               */
/*-------------------------------------------------------------------------------------------*/

cellule_t ** RECH_PREC(int date_deb, cellule_t ** t)
{
    cellule_t * cour = *t;
    cellule_t ** prec = t;
    while ((cour != NULL) && (cour->deb_valid < date_deb))
    {
        prec = &(cour->suivant);
        cour = cour->suivant;
    }
    return prec;
}


/*-------------------------------------------------------------------------------------------*/
/* ADJ-CELL             Adjonction d'une cellule dans une liste chaînée                      */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                          - Si la cellule à insérer existe :                               */
/*                              - Faire pointer la nouvelle cellule sur la cellule suivante  */
/*                              - Faire pointer la cellule précédente sur la nouvelle cellule*/
/*                          FIN                                                              */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : prec        Pointeur sur le précédent                                        */
/*              nouv        Pointeur sur la cellule à insérer                                */
/*-------------------------------------------------------------------------------------------*/

void ADJ_CELL(cellule_t ** prec, cellule_t * nouv)
{
    if (nouv != NULL)
    {
        nouv->suivant = *prec;
        *prec = nouv;
    }
}
```

```
/*-------------------------------------------------------------------------------------------*/
/* INSERT_CELL          Insertion d'une cellule dans une liste chaînée triée                 */
/*                      (selon la date de début de validité)                                 */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                           - Rechercher l'adresse à laquelle il faut insérer la nouvelle   */
/*                             cellule afin de garder la liste chaînée triée                 */
/*                             (sous-programme RECH_PREC)                                    */
/*                           - Insérer la nouvelle cellule à cette adresse                   */
/*                             (sous-programme ADJ_CEL)                                      */
/*                           FIN                                                             */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : nouv         Pointeur sur la cellule à insérer                               */
/*              t            Pointeur sur le pointeur de tête de la liste chaînée            */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              prec         Pointeur sur le précédent                                       */
/*-------------------------------------------------------------------------------------------*/

void INSERT_CELL(cellule_t * nouv, cellule_t ** t)
{
    cellule_t ** prec;
    prec = RECH_PREC(nouv->deb_valid, t);
    ADJ_CELL(prec, nouv);
}
```

```
/*-------------------------------------------------------------------------------------------*/
/* SUPP_MESS_EXP               Supprime les messages devenus obsolètes dans une              */
/*                             liste chaînée                                                 */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                           - Sauvegarder la date système dans une variable                 */
/*                             (sous-programme DATE_DU_JOUR)                                 */
/*                           - Tant que l'on n'a pas atteint la fin de la liste chaînée :    */
/*                               - Si la date système est supérieure à la date de fin de     */
/*                                 validité de la cellule courante :                         */
/*                                   - Supprimer la cellule courante (sous-programme SUPP_CELL)  */
/*                               - Sinon : Avancer la pointeur sur le précédent dans la liste */
/*                           - Avancer le pointeur courant dans la liste                     */
/*                           FIN                                                             */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : t               Pointeur sur le pointeur de tête de la liste                 */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              date            Entier représentant la date du jour                          */
/*              prec            Pointeur sur le précédant de la cellule courante             */
/*              cour            Pointeur sur la cellule courante                             */
/*-------------------------------------------------------------------------------------------*/
```

```c
/*-------------------------------------------------------------------------------------------*/
/* SUPP_CELL                    Supprime une cellule dans une liste chaînée                  */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                          - Faire pointer la cellule précédente vers la cellule suivante   */
/*                          - Libérer la cellule courante                                    */
/*                          FIN                                                              */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : prec            Pointeur sur le précédent de la cellule à supprimer          */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              cour            Pointeur sur la cellule à supprimer                          */
/*-------------------------------------------------------------------------------------------*/

void SUPP_CELL(cellule_t ** prec)
{
    cellule_t * cour = *prec;
    *prec = cour->suivant;
    free(cour);
}
```

```c
/*-------------------------------------------------------------------------------------------*/
/* SUPP_LCH                Libère une liste chaînée                                          */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                          - Tant que le pointeur de tête n'est pas NULL :                  */
/*                              - Libérer la première cellule                                */
/*                          FIN                                                              */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : t               Pointeur de tête de la liste chaînée                         */
/*-------------------------------------------------------------------------------------------*/

void SUPP_LCH(cellule_t **t)
{
    cellule_t **prec = t;
    while(*prec != NULL)
    {
        SUPP_CELL(prec);
    }
}
```

2.2 Programme de gestion de messages dans TP1.c

2.2.1 Création d'une liste chaînée triée à partir d'un fichier d'entrée

```
/*-------------------------------------------------------------------------------------------*/
/* LECTURE                      Création d'une liste chaînée à partir du                     */
/*                              fichier d'entrée                                             */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                           - Allouer un bloc de 100 mots                                   */
/*                             (bloc temporaire pour stocker une chaîne de caractères)       */
/*                           - Si l'allocation est réussie :                                 */
/*                               - Ouvrir le fichier en lecture                              */
/*                               - Si le fichier est ouvert :                                */
/*                                   - Tant que l'on n'a pas atteint la fin du fichier :     */
/*                                       - Créer une nouvelle cellule                        */
/*                                         (sous-programme CREER_CELL)                       */
/*                                       - Copier les valeurs lues dans le fichier dans      */
/*                                         cette cellule                                     */
/*                                       - Insérer cette cellule dans la liste chaînée       */
/*                                         (sous-programme INSERT_CELL)                      */
/*                                   - Fermer le fichier                                     */
/*                                   - Libérer le bloc de caractères temporaire              */
/*                               - Sinon : Afficher un message d'erreur                      */
/*                           - Sinon : Afficher un message d'erreur                          */
/*                           FIN                                                             */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : nom_fichier     Nom du fichier d'entrée (chaîne de caractères)               */
/*              t               Pointeur sur le pointeur de tête de la liste chaînée         */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              fichier         Pointeur sur le fichier d'entrée                             */
/*              deb, fin        Deux entiers représentant respectivement les dates de        */
/*                              début et fin de validité du message                          */
/*              nouv            Pointeur sur la cellule à insérer dans la liste chaînée      */
/*              txt             Chaîne de caractères intermédiaire contenant le message      */
/*              texte           Chaîne de caractères contenant le message                    */
/*                                                                                           */
/*  En sortie : erreur          Entier valant 0 si le fichier est lu et 1 sinon              */
/*-------------------------------------------------------------------------------------------*/
```

```c
int LECTURE(char * nom_fichier, cellule_t ** t)
{
        FILE * fichier;
        int deb, fin, erreur = 1;
        cellule_t * nouv;
        char * txt, * texte;
        txt = (char *)malloc(100*sizeof(char));
        if (txt != NULL)
        {
            fichier = fopen(nom_fichier, "r");
            if (fichier != NULL)
            {
                erreur = 0;
                while (fscanf(fichier, "%d %d", &deb, &fin) != EOF)
                {
                    fgets(txt, 100, fichier);
                    texte = (char *)malloc(strlen(txt)*sizeof(char));
                    strcpy(texte, txt);
                    texte[strlen(txt)-1] = '\0';
                    nouv = CREER_CELL(deb, fin, texte);
                    INSERT_CELL(nouv, t);
                }
                fclose(fichier);
                free(txt);
            }
            else
                printf("Erreur lors de l'ouverture du fichier\n");
        }
        else
            printf("Erreur : problème d'allocation\n");

        return erreur;
}
```

2.2.2 Création d'un fichier à partir d'une liste chaînée triée

```
/*-------------------------------------------------------------------------------------------*/
/* SAUVEGARDE                  Sauvegarde de la liste chaînée d'entrée dans un fichier       */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                          - Lire le nom du fichier de sauvegarde choisi par l'utilisateur  */
/*                          - Allouer un bloc de la taille de ce nom de fichier              */
/*                          - Si l'allocation est réussie :                                  */
/*                              - Ouvrir un fichier en écriture portant ce nom               */
/*                              - Si le fichier est ouvert :                                 */
/*                                  - Tant que l'on n'a pas atteint la fin du fichier :      */
/*                                      - Copier dans le fichier (sur une ligne) les valeurs */
/*                                        contenues dans la cellule courante                 */
/*                                      - Avancer le pointeur courant dans la liste chaînée  */
/*                                  - Fermer le fichier                                      */
/*                              - Sinon : Afficher un message d'erreur                       */
/*                          - Sinon : Afficher un message d'erreur                           */
/*                          FIN                                                              */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : t               Pointeur sur le pointeur de tête de la liste                 */
/*              nom_fichier     Pointeur sur une chaîne de caractères désignant le nom du    */
/*                              du fichier à créer                                           */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              fichier         Pointeur sur le fichier d'entrée                             */
/*              cour            Pointeur sur la cellule courante                             */
/*                                                                                           */
/*  En sortie : erreur          Entier valant 0 si le fichier est créé et 1 sinon            */
/*-------------------------------------------------------------------------------------------*/
```

```c
int SAUVEGARDE(cellule_t * t, char * nom_fichier)
{
    FILE * fichier;
    cellule_t * cour = t;
    int erreur = 1;
    char buffer [20];
    printf("Nom du fichier de sauvegarde :\n");
    scanf("%s", buffer);
    nom_fichier = (char *)malloc(sizeof(char)*strlen(buffer));
    if (nom_fichier != NULL)
    {
        strcpy(nom_fichier, buffer);
        fichier = fopen(nom_fichier, "w");
        if ((fichier != NULL))
        {
            erreur = 0;
            while (cour != NULL)
            {
                fprintf(fichier, "%d %d %s\n", cour->deb_valid, cour->fin_valid, cour->texte);
                cour = cour->suivant;
            }
            fclose(fichier);
        }
        else
            printf("Erreur lors de l'ouverture du fichier\n");
    }
    else
        printf("Erreur : problème d'allocation\n");
    return erreur;
}
```

2.2.3 La date du jour

```
/*-------------------------------------------------------------------------------------------*/
/* DATE_DU_JOUR                Retourne un entier représentant la date du jour               */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                          - Récupérer la date système avec la fonction localtime()         */
/*                          - Mettre cette date de la forme aaaammjj                         */
/*                          FIN                                                              */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              cour            Pointeur sur la cellule courante                             */
/*              date            Entier représentant la date du jour                          */
/*                                                                                           */
/*  En sortie : date            Entier représentant la date du jour sous la forme            */
/*                              aaaammjj                                                     */
/*-------------------------------------------------------------------------------------------*/

int DATE_DU_JOUR()
{
    time_t temps;
    struct tm * ajd;
    int date = 0;

    temps = time(NULL);
    ajd = localtime(&temps);

    int annee = ajd->tm_year + 1900;            /* tm_year : annee courante - 1900 */
    int mois = ajd->tm_mon + 1;                 /* tm_mon : mois [0-11] */
    int jour = ajd->tm_mday;                    /* tm_mday : jour du mois [1-31] */

    date += annee*10000 + mois*100 + jour;      /* date de la forme aaaammjj */
    return date;
}
```

2.2.4 Suppression des messages obsolètes

```
/*-------------------------------------------------------------------------------------------*/
/* AFF_NON_EXP                  Affiche les messages non expirés à la date du jour           */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                           - Sauvegarder dans une variable la date système                 */
/*                             (sous-programme DATE_DU_JOUR)                                 */
/*                           - Tant que l'on n'a pas atteint la fin de la liste chaînée :    */
/*                               - Si la date système est inférieure à la date de fin de     */
/*                                 validité de la cellule courante :                         */
/*                                   - Afficher les valeurs contenues dans la cellule courante */
/*                               - Avancer le pointeur courant dans la liste chaînée         */
/*                           FIN                                                             */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : t               Pointeur sur le pointeur de tête de la liste                 */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              cour            Pointeur sur la cellule courante                             */
/*              date            Entier représentant la date du jour                          */
/*-------------------------------------------------------------------------------------------*/

void AFF_NON_EXP(cellule_t ** t)
{
    cellule_t * cour = *t;
    int date;
    date = DATE_DU_JOUR();
    printf("date du jour : %d\n", date);
    printf("Messages non expirés :\n")
    while (cour != NULL)
    {
        if (date < cour->fin_valid)
            printf("%d %d %s\n", cour->deb_valid, cour->fin_valid, cour->texte);
        cour = cour->suivant;
    }
}
```

2.2.5 Suppression des messages obsolètes

```
/*-------------------------------------------------------------------------------------------*/
/* SUPP_MESS_EXP                Supprime les messages devenus obsolètes dans une             */
/*                              liste chaînée                                                */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                           - Sauvegarder la date système dans une variable                 */
/*                             (sous-programme DATE_DU_JOUR)                                 */
/*                           - Tant que l'on n'a pas atteint la fin de la liste chaînée :    */
/*                               - Si la date système est supérieure à la date de fin de     */
/*                                 validité de la cellule courante :                         */
/*                                   - Supprimer la cellule courante (sous-programme SUPP_CELL)  */
/*                               - Sinon : Avancer la pointeur sur le précédent dans la liste */
/*                           - Avancer le pointeur courant dans la liste                     */
/*                           FIN                                                             */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : t               Pointeur sur le pointeur de tête de la liste                 */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              date            Entier représentant la date du jour                          */
/*              prec            Pointeur sur le précédant de la cellule courante             */
/*              cour            Pointeur sur la cellule courante                             */
/*-------------------------------------------------------------------------------------------*/
```

```
void SUPP_MESS_EXP(cellule_t ** t)
{
    int date = DATE_DU_JOUR();
    cellule_t ** prec = t;
    cellule_t * cour = *t;
    while (cour != NULL)
    {
        if (date > cour->fin_valid)
            SUPP_CELL(prec);
        else
            prec = &cour->suivant;
        cour = *prec;
    }
}
```

2.2.6 Modification de la date de début de messages connaissant leur date de début initiale

```
/*-------------------------------------------------------------------------------------------*/
/* MODIF_DATE_DEB              Modifie la date de début des messages d'une date              */
/*                             initiale donnée                                               */
/*                                                                                           */
/*  Algorithme de principe:                                                                  */
/*                          - Tant que l'on n'a pas atteint la fin de la liste chaînée :     */
/*                              - Si la date de début de validité de la cellule courante est */
/*                                égale à la date initiale à modifier :                      */
/*                                  - Modifier la date de début de validité de la cellule en */
/*                                    la date voulue                                         */
/*                              - Avancer le pointeur courant dans la liste chaînée          */
/*                          FIN                                                              */
/*                                                                                           */
/*  Lexique                                                                                  */
/*                                                                                           */
/*  En entrée : t              Pointeur de tête de la liste chaînée                          */
/*              date_init      Entier représentant la date initiale à modifier               */
/*              date_nouv      Entier représentant la nouvelle date                          */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              cour           Pointeur sur la cellule courante                              */
/*-------------------------------------------------------------------------------------------*/

void MODIF_DATE_DEB(cellule_t * t, int date_init, int date_nouv)
{
    cellule_t * cour = t;
    while (cour != NULL)
    {
        if ((cour->deb_valid == date_init))
            cour->deb_valid = date_nouv;
        cour = cour->suivant;
    }
}
```

2.3 Programme principal (Main.c)

```
/*-------------------------------------------------------------------------------------------*/
/*  Lexique                                                                                  */
/*                                                                                           */
/*  Variables intermédiaires :                                                               */
/*              okL1,okL2,okL3  Entiers valant 1 s'il y a uune erreur dans la création de la */
/*                              liste chaînée pour les différents tests de la fonction lecture*/
/*              okS1,okS2       Entiers valant 1 s'il y a uune erreur dans la sauvegarde de la*/
/*                              liste chaînée pour les différents tests de la fonction sauvegarde*/
/*              tete            Pointeur de tête de la liste chaînée pour le cas où le fichier*/
/*                              existe                                                       */
/*              tete_vide       Pointeur de tête de la liste chaînée pour le cas où le fichier*/
/*                              n'existe pas                                                 */
/*              tete_test       Pointeur de tête de la liste chaînée pour le cas où le fichier*/
/*                              est vide                                                     */
/*              nom_fichier     Chaîne de caractères tapé au clavier dans la fonction sauvegarde*/
/*-------------------------------------------------------------------------------------------*/
```

```c
int main()
{
    int okL1 = 1, okL2 = 1, okL3 = 1;
    int okS1 = 1, okS2 = 1;
    cellule_t * tete = NULL;
    cellule_t * tete_test = NULL;
    cellule_t * tete_vide = NULL;
    char * nom_fichier;

    /* Test de la lecture du fichier */
        /*fichier existant*/
    okL1 = LECTURE("messages.txt", &tete);
    printf("Test de la fonction de lecture de fichier : ");
    if (okL1 == 1)
        printf("Erreur : fichier non lu \n\n");
    else
        printf("Fichier lu, liste chainee triée \n\n");
        /*fichier qui n'existe pas*/
    okL2 = LECTURE("messagges.txt", &tete_test);
    printf("Test de la fonction de lecture de fichier : ");
    if (okL2 == 1)
        printf("Erreur : fichier non lu \n\n");
    else
        printf("Fichier lu, liste chainee triée \n\n");
        /*fichier vide*/
    okL3 = LECTURE("vide.txt", &tete_vide);
    printf("Test de la fonction de lecture de fichier : ");
    if (okL3 == 1)
        printf("Erreur : fichier non lu \n\n");
    else
        printf("Fichier lu, liste chainee triée \n\n");
```

```c
    /*Test de la sauvegarde de la liste chaînée dans un fichier */
            /*liste chaînée existante*/
    okS1 = SAUVEGARDE(tete, nom_fichier);
    printf("\nTest de la fonction de sauvegarde de la liste chaînée : ");
    if (okS1 == 1)
        printf("Erreur : fichier non crée \n\n");
    else
        printf("Fichier crée \n\n");
            /*liste chaînée vide*/
    okS2 = SAUVEGARDE(tete_vide, nom_fichier);
    printf("\nTest de la fonction de sauvegarde de la liste chaînée : ");
    if (okS2 == 1)
        printf("Erreur : fichier non crée \n\n");
    else
        printf("Fichier crée \n\n");

    /* Test de la fonction d'affichage des messages non expirés */
    AFF_NON_EXP(&tete);

    /* Test de la fonction de suppression des messages expirés */
    printf("\nTest de la fonction de suppression des messages expires\n\n");
    SUPP_MESS_EXP(&tete);
    printf("Messages expirés supprimés\n");
    SAUVEGARDE(tete, nom_fichier);

    /* Test de la fonction de modification d'une date de début connue */
            /*date deb_valid inexistante*/
    printf("\nTest de la fonction de modification d'une date de début connue :\n");
    MODIF_DATE_DEB(tete, 19980406, 19980608);
    printf("Date modifiée\n");
    SAUVEGARDE(tete, nom_fichier);
            /*date deb_valid existante*/
    printf("\nTest de la fonction de modification d'une date de début connue :\n");
    MODIF_DATE_DEB(tete, 19980405, 19980608);
    printf("Date modifiée\n");
    SAUVEGARDE(tete, nom_fichier);

    /*Libération de la liste chaînée*/
    SUPP_LCH(&tete);

    return 0;
}
```

Partie 3

Compte rendu d'éxucution

3.1 Makefile

```
# compilateur
CC = gcc
# options
CFLAGS = -Wall -Wextra -g
LDFLAGS =
# liste des fichiers objets
OBJ = TP1.o Lch.o Main.o
# règle de production finale tp :

TP1 : $(OBJ)
    $(CC) $(OBJ) $(LDFLAGS) -o TP1
    @echo "Lancer le programme avec ./TP1"

# règle de production pour chaque fichier
TP1.o : TP1.h Lch.h TP1.c
    $(CC) -c TP1.c $(CFLAGS)
Lch.o : Lch.h Lch.c
    $(CC) -c Lch.c $(CFLAGS)
Main.o : TP1.h Lch.h TP1.c Lch.c
    $(CC) -c Main.c $(CFLAGS)

clean :
        rm $(OBJ)
```

3.2 Jeux de test

Les tests des fonctions ont été effectués sur différents fichiers d'entrée : un fichier inexistant (messagges.txt), un fichier vide (vide.txt) et un fichier normal (messages.txt).

Les résultats des différents tests sont observés à travers le terminal, les fichiers crées, ainsi que le débogueur ddd pour l'affichage de liste chaînées.

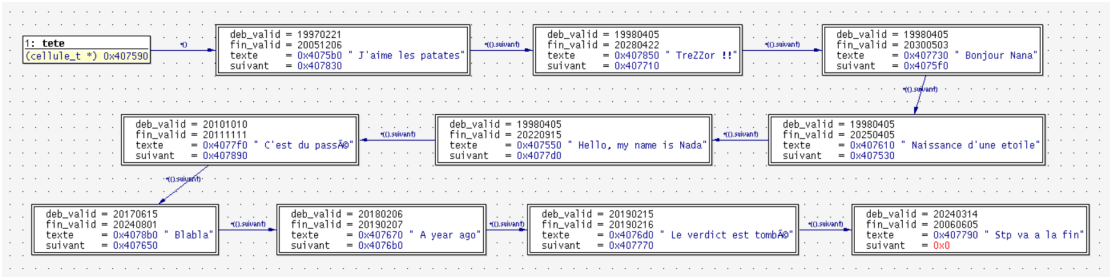
Le fichier d'entrée messages.txt utilisé pour les tests se présente comme suit :

messages.txt

```
19980405 20220915 Hello, my name is Nada
19970221 20051206 J'aime les patates
19980405 20250405 Naissance d'une etoile
20180206 20190207 A year ago
20190215 20190216 Le verdict est tombé
19980405 20300503 Bonjour Nana
20240314 20060605 Stp va a la fin
20101010 20111111 C'est du passé
19980405 20280422 TreZZor !!
20170615 20240801 Blabla
```

- Création d'une liste chaînée avec insertion en tête, insertion au milieu et insertion en fin

```
gcc TP1.o Lch.o Main.o  -o TP1
Lancer le programme avec ./TP1
[naboutadgh@etud TP1.14.03]$ ./TP1
Test de la fonction de lecture de fichier : Fichier lu, liste chainee triée
```

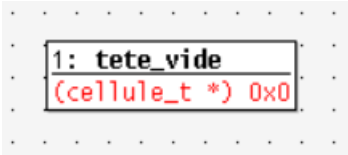


- Création liste chaînée à partir d'un fichier inexistant

```
Erreur lors de l'ouverture du fichier
```

- Création liste chaînée à partir d'un fichier vide

```
Test de la fonction de lecture de fichier : Fichier lu, liste chainee triée
```



- Création d'un fichier à partir d'une liste chaînée existante

```
Nom du fichier de sauvegarde :
save
```

save

```
1   19970221 20051206  J'aime les patates
2   19980405 20280422  TreZZor !!
3   19980405 20300503  Bonjour Nana
4   19980405 20250405  Naissance d'une etoile
5   19980405 20220915  Hello, my name is Nada
6   20101010 20111111  C'est du passé
7   20170615 20240801  Blabla
8   20180206 20190207  A year ago
9   20190215 20190216  Le verdict est tombé
10  20240314 20060605  Stp va a la fin
```

- Création d'un fichier à partir d'une liste chaînée vide

```
Test de la fonction de sauvegarde de la liste chaînée : Fichier crée

Nom du fichier de sauvegarde :
save2
```

save2

```
1
```

- Affichage des messages non expirés

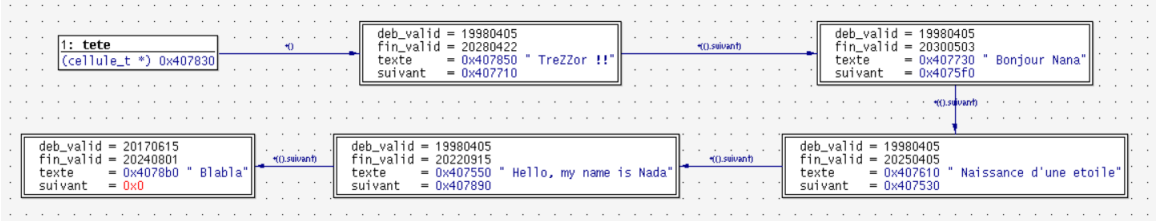
```
Test de la fonction de sauvegarde de la liste chaînée : Fichier crée

date du jour : 20190314
Messages non expirés :
19980405 20280422  TreZZor !!
19980405 20300503  Bonjour Nana
19980405 20250405  Naissance d'une etoile
19980405 20220915  Hello, my name is Nada
20170615 20240801  Blabla
```

- Suppression d'un message obsolète en tête de la liste chaînée : "J'aime les patates"
- Suppression d'un message obsolète dans la liste chaînée : "C'est du passé"
- Suppression d'un message obsolète en fin de la liste chaînée : "Stp va a la fin"

```
Test de la fonction de suppression des messages expires

Messages expirés supprimés
Nom du fichier de sauvegarde :
messexp
```



messexp

```
19980405 20280422  TreZZor !!
19980405 20300503  Bonjour Nana
19980405 20250405  Naissance d'une etoile
19980405 20220915  Hello, my name is Nada
20170615 20240801  Blabla
```

- Modifier la date de début de validité d'une date non présente dans le fichier d'entrée : 19980406

```
Test de la fonction de modification d'une date de début connue :
Date modifiée
Nom du fichier de sauvegarde :
modifdate
```

modifdate

```
19980405 20280422  TreZZor !!
19980405 20300503  Bonjour Nana
19980405 20250405  Naissance d'une etoile
19980405 20220915  Hello, my name is Nada
20170615 20240801  Blabla
```

- Modifier la date de début de validité d'une date présente dans le fichier d'entrée : 19980405

```
Test de la fonction de modification d'une date de début connue :
Date modifiée
Nom du fichier de sauvegarde :
modifdate2
```

modifdate2

```
19980608 20280422  TreZZor !!
19980608 20300503  Bonjour Nana
19980608 20250405  Naissance d'une etoile
19980608 20220915  Hello, my name is Nada
20170615 20240801  Blabla
```

- Libération de la liste chaînée