Bousquet Edith

Davies Rhys

Groupe : G31

ZZ1

TP n°1 de SDD

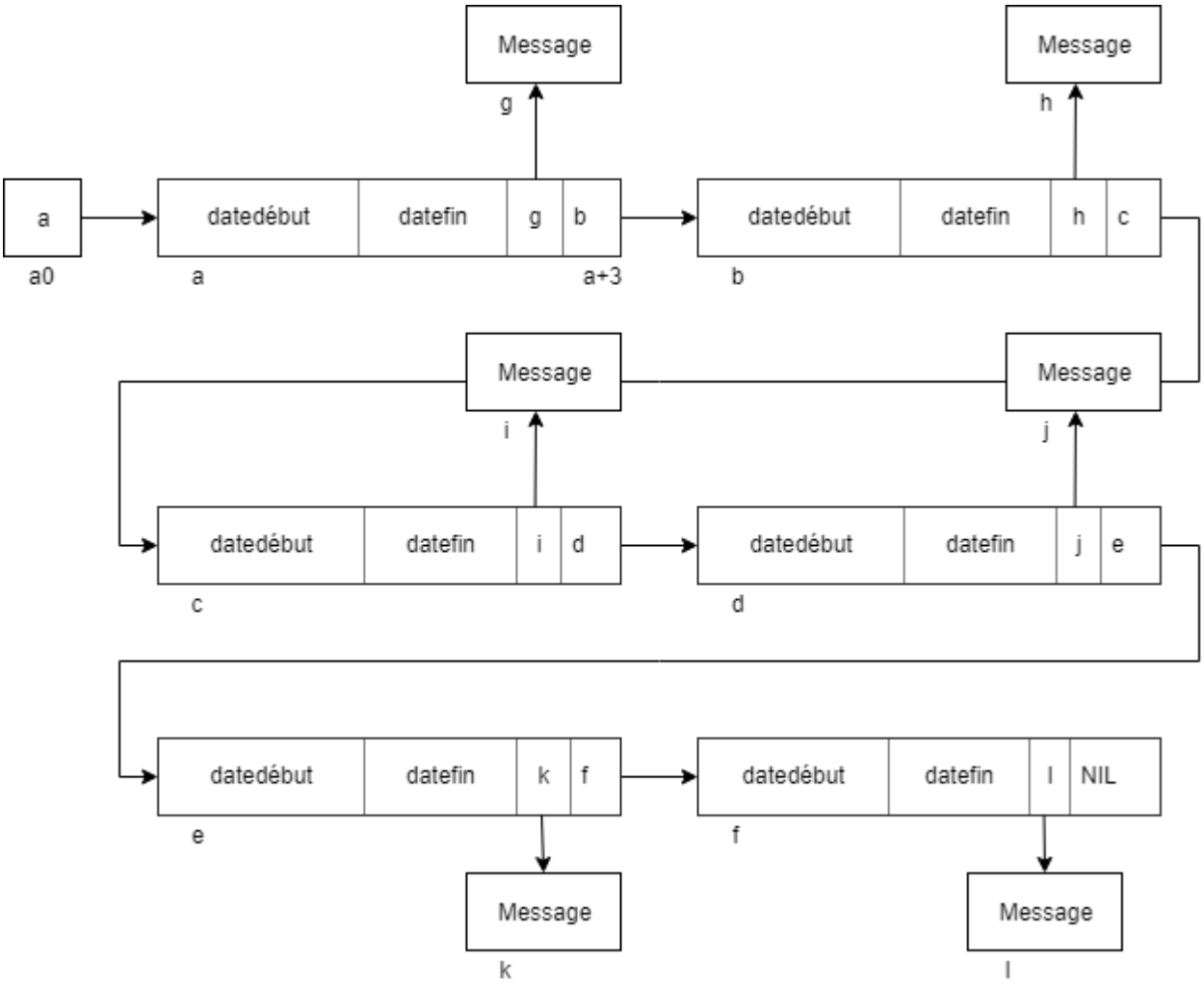
ISIMA
Institut Supérieur d'Informatique de Modélisation et de leurs Applications

Année 2018/2019

Sommaire

Structure

Schéma



Description

La structure se compose :

- Un entier qui est la date de début de la forme aaaammjj
- Un entier qui est la date de fin de la forme aaaammjj
- Un pointeur sur une chaine de caractère qui est le message

Organisation du code

Nous avons organisé notre code en trois fichiers :

- main.c : Dans ce fichier se trouve le programme principal
- tp1.h : Dans ce fichier se trouve les déclarations des fonctions. Au-dessus de chaque fonction nous avons placé les entêtes avec une petite description de la fonction, les entrées et les sorties.
- tp1.c : Dans ce fichier se trouve les définitions des fonctions. Les fonctions sont commentées et au-dessus de chaque fonction se trouve une courte description.

Nous avons créé un Makefile. Il nous permet de compiler l'ensemble du code et de créer l'exécutable.

Le fichier contenant les données se nomme remplissage.txt. Pour les différents cas, nous avons créé un fichier par cas identifié :

- remplissage.txt : Fichier général
- remplissageDdEgaux.txt : Fichier avec les dates de début qui sont égales
- remplissageDfinDd.txt : Fichier avec les dates de fin qui sont inférieures aux dates de début
- remplissageMsgSup100.txt : Fichier avec un message de plus de 100 caractères
- remplissageVide.txt : Fichier vide

Dans le code, nous créons des fichiers de sortie :

- sortieLChTriee.txt : Ce fichier contient la liste chainée triée
- sortieLChDateObs.txt : Ce fichier contient la liste chainée triée sans les dates de débuts qui sont obsolètes
- sortieLChDateModifier.txt : Ce fichier contient la liste chainée triée avec des dates de début qui ont été modifié

Pour insérer la date à modifier dans le programme, on peut les insérer directement depuis le terminal ou on peut créer un fichier contenant les dates. Nous l'avons appelé parModifDate.txt.

Code

Dans cette partie se trouve les captures d'écrans de notre code.

tp1.h

Dans ce fichier se trouve les déclarations des fonctions. Au-dessus de chaque fonction nous avons placé les entêtes avec une petite description de la fonction, les entrées et les sorties.

```c
/* ------------------------------------------------------------------------------------ */
/*                                                                                      */
/*  tp1.h   Contient toutes les déclarations de fonctions, la déclaration de la structure  */
/*                                                                                      */
/* ------------------------------------------------------------------------------------*/

#ifndef tp1_h
#define tp1_h
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

/*Déclarartion de la cellule*/
typedef struct Cellule
{
    int               dd;       /*Date de début*/
    int               df;       /*Date de fin*/
    char            * msg;      /*Pointeur sur le message*/
    struct Cellule  * ptsuiv;   /*Pointeur sur la cellule suivant*/
}Cellule_t;

/* ------------------------------------------------------------------- */
/* InitLch         Permet l'allocation d'une nouvelle cellule          */
/*                                                                     */
/* En entrée: Rien                                                     */
/*                                                                     */
/* En sortie:  L'adresse de la cellule                                 */
/* ------------------------------------------------------------------- */
Cellule_t * InitLch();

/* ------------------------------------------------------------------- */
/* CreerCellule         Permet la création de la cellule               */
/*                                                                     */
/* En entrée: Deux entiers (date de début pdd date de fin pdf) une     */
/* chaine                                                              */
/* de  cararctere (message pmsg)                                       */
/*                                                                     */
/* En sortie:  L'adresse de la cellule                                 */
/* ------------------------------------------------------------------- */
Cellule_t * CreerCellule(int, int, char []);
```

```c
/* ------------------------------------------------------------------ */
/* RecherchePrecDate   Recherche dans la liste chainée de la cellule  */
/* contenant une date supérieure à celle passée en paramétre          */
/*                                                                    */
/* En entrée: Un pointeur sur le pointeur de tête (a0) et la date à   */
/* comparer                                                           */
/*                                                                    */
/* En sortie:  L'adresse du pointeur pécédent                         */
/* ------------------------------------------------------------------ */
Cellule_t ** RecherchePrecDate(Cellule_t ** , int );

/* ------------------------------------------------------------------ */
/* InsererCellule   Permet d'insérer une cellule dans une liste chainée */
/* triée                                                              */
/*                                                                    */
/* En entrée: Un pointeur sur le pointeur de tête (a0) et le pointeur */
/* de la nouvelle cellule à insérer                                   */
/*                                                                    */
/* En sortie:  Le pointeur sur le pointeur de tête (a0)               */
/* ------------------------------------------------------------------ */
void InsererCellule(Cellule_t ** , Cellule_t * );

/* ------------------------------------------------------------------ */
/* Afficher   Permet l'affichage de la liste chainée                  */
/*                                                                    */
/* En entrée: Le pointeur de tête (a0) et le fichier où               */
/* il doit être affiché                                               */
/*                                                                    */
/* En sortie: Le fichier                                              */
/* ------------------------------------------------------------------ */
void Afficher(Cellule_t *, FILE*);

/* ------------------------------------------------------------------ */
/* Ecrire   Permet l'écrire de la liste chainée dans le fichier dont le */
/* nom est passé en paramètre                                         */
/*                                                                    */
/* En entrée: Le pointeur de tête (a0) et le nom du fichier où  on doit */
/* enregistrer la liste chainée                                       */
/*                                                                    */
/* En sortie: Le code pour savoir si l'écriture c'est bien passée     */
/* ------------------------------------------------------------------ */
int Ecrire(Cellule_t *, char *);
```

```
86  /* ------------------------------------------------------------------ */
87  /* RecupererDateCour   Permet de récupérer la date du jour sous la forme*/
88  /* aaaammjj                                                           */
89  /*                                                                    */
90  /* En entrée: Rien                                                    */
91  /*                                                                    */
92  /* En sortie: la date du jour                                         */
93  /* ------------------------------------------------------------------ */
94  int RecupererDateCour();
95
96  /* ------------------------------------------------------------------ */
97  /* AfficherControleDate   Permet l'affichage de la liste chainée triée  */
98  /* sans les dates obsolètes                                           */
99  /*                                                                    */
100 /* En entrée: Le pointeur de tête (a0) et et le fichier où            */
101 /* il doit être affiché                                               */
102 /*                                                                    */
103 /* En sortie: Le fichier                                              */
104 /* ------------------------------------------------------------------ */
105 void AfficherControleDate(Cellule_t *, FILE*);
106
107 /* ------------------------------------------------------------------ */
108 /* SuppEltLCh   Permet de supprimer un élément de la liste chainée    */
109 /*                                                                    */
110 /* En entrée: L'adresse du pointeur précédent                         */
111 /*                                                                    */
112 /* En sortie: L'adresse du pointeur précédent                         */
113 /* ------------------------------------------------------------------ */
114 void SuppEltLCh(Cellule_t **);
115
116 /* ------------------------------------------------------------------ */
117 /* SuppDateObs   Permet de supprimer les éléments dans la liste chainée */
118 /* dont la date de début est obsolète                                 */
119 /*                                                                    */
120 /* En entrée: Un pointeur sur le pointeur de tête (a0)                */
121 /*                                                                    */
122 /* En sortie: Le pointeur sur le pointeur de tête (a0)                */
123 /* ------------------------------------------------------------------ */
124 void SuppDateObs(Cellule_t **);
```

```
126  /* ------------------------------------------------------------------ */
127  /* ModifierDateDebut   Permet de modifier la date de début des cellules */
128  /* ayant une certaine date de début                                   */
129  /*                                                                    */
130  /* En entrée: Un pointeur sur le pointeur de tête (a0), deux entiers, la*/
131  /*  date de début à modifier et la nouvelle date                      */
132  /*                                                                    */
133  /* En sortie: Le pointeur sur le pointeur de tête (a0)                */
134  /* ------------------------------------------------------------------ */
135  void ModifierDateDebut(Cellule_t **, int , int);
136
137  /* ------------------------------------------------------------------ */
138  /* ValiditeDate   Permet de vérifier si une date de début est bien    */
139  /* inférieure à la date de fin                                        */
140  /*                                                                    */
141  /* En entrée: Deux entiers, la date de début et la date de fin        */
142  /*                                                                    */
143  /* En sortie: Un entier, 0 si vrai et 1 si faux                       */
144  /* ------------------------------------------------------------------ */
145  int ValiditeDate(int, int);
146
147  /* ------------------------------------------------------------------ */
148  /* Liberer        Permet de libérer la liste de chainée à la fin du   */
149  /* programme                                                          */
150  /*                                                                    */
151  /* En entrée: Un pointeur sur le pointeur de tête (a0)                */
152  /*                                                                    */
153  /* En sortie: Le pointeur de tête                                     */
154  /* ------------------------------------------------------------------ */
155  void Liberer(Cellule_t * * );
156  #endif
157
```

tp1.c

Dans ce fichier se trouve les définitions des fonctions. Les fonctions sont commentées et au-dessus de chaque fonction se trouve une courte description.

Pour chaque fonction, nous avons détaillé le principe de la fonction, les entrées et les sorties et le lexique.

InitLch :

- Principe : Fonction permettant l'allocation d'une nouvelle cellule
- Entrée : Pas d'entrée
- Sortie : L'adresse de la cellule allouée.
- Lexique :

  Pas de variable.

```
/*Fonction permettant l'allocation d'une nouvelle cellule*/
/*Renvoie l'adresse de la cellule*/
Cellule_t * InitLch()
{
    return((Cellule_t  *) malloc(sizeof(Cellule_t)));
}
```

CreerCellule :

- Principe : Fonction permettant la création de la cellule
- Entrée : Deux entiers, la date de début (pdd) et la date de fin (pdf), et une chaine de caractère, le message (pmsg)
- Sortie : L'adresse de la nouvelle cellule
- Lexique :

  Entrée :

  - pdd : variable contenant la date de début à insérer
  - pdf : variable contenant la date de fin à insérer
  - pmsg : variable contenant le message à insérer

  Sortie :

  - c : adresse de la cellule alloue

```
/*Fonction permettant la création de la cellule*/
/*Renvoie l'adresse de la cellule*/
Cellule_t * CreerCellule(int pdd, int pdf, char pmsg[])
{
    /*Allocation de la cellule*/
    Cellule_t * C = InitLch();

    /*Vérification que la cellule a bien été allouée*/
    if(C)
    {
        /*Insertion des données dans la cellule*/
        C->dd = pdd;
        C->df = pdf;
        C->msg = (char*) malloc(sizeof(char)*(((int) strlen(pmsg)) +1));
        if(C->msg)
        {
            strcpy(C->msg, pmsg);
        }
        C->ptsuiv = NULL;
    }

    return (C);
}
```

RecherchePrecDate :

- Principe : Fonction de recherche dans la liste chainée de la cellule contenant une date supérieure à celle passée en paramètre
- Entrée : Un pointeur sur le pointeur de tête (a0) et un entier, la date à comparer (date)
- Sortie : L'adresse du pointeur précédent
- Lexique :

  Entrée :

  - a0 : adresse du pointeur de tête
  - date : variable contenant la date à comparer

  Sortie :

  - prec : adresse de pointeur de parcours

```c
/*Fonction de recherche dans la liste chainée de la cellule contenant une date supérieure à celle passée en paramétre*/
/*Renvoie l'adresse du précedent */
Cellule_t ** RecherchePrecDate(Cellule_t ** a0, int date)
{
    /*Initialisation du pointeur précedent sur le pointeur de tête*/
    Cellule_t ** prec = a0;

    /*Boucle tant qu'on est pas arrivé à la fin de la liste et que la date est supérieure à celle dans la liste*/
    while ((*prec) && ((*prec)->dd < date))
    {
        /*Incrémentation du pointeur précedent*/
        prec = &((*prec)->ptsuiv);
    }

    return (prec);
}
```

InsererCellule :

- Principe : Procédure permettant d'insérer une cellule dans une liste chainée triée
- Entrée : Un pointeur sur le pointeur de tête (a0) et l'adresse de la cellule à insérer (nv)
- Sortie : Le pointeur de tête
- Lexique :

  Entrée :

  - nv : adresse de la cellule à insérer

  Entrée / Sortie :

  - a0 : adresse du pointeur de tête

  Intermédiaire :

  - prec : adresse du pointeur précédent où on doit insérer la nouvelle cellule.

```c
/*Procédure permettant d'insérer une cellule dans une liste chainée triée*/
void InsererCellule(Cellule_t ** a0, Cellule_t * nv)
{
    /*Initialisation du pointeur précédent à l'adresse où on doit insérer le nouvelle élément*/
    Cellule_t ** prec = RecherchePrecDate(a0, nv->dd);
    nv->ptsuiv = (*prec);
    (*prec)= nv;

}
```

Afficher :

- Principe : Procédure permettant l'affichage de la liste chainée
- Entrée : Le pointeur de tête (a0) et le fichier où la liste chainée doit être affichée
- Sortie : Le fichier
- Lexique :

  Entrée :

  - a0 : adresse de la première cellule de la liste chainée

  Entrée / Sortie :

  - fich : adresse du fichier où la liste sera affiché

  Intermédiaire :

  - cour : adresse de l'élément courant

```c
/*Procédure permettant l'affichage de la liste chainée*/
void Afficher(Cellule_t * a0, FILE* fich)
{
    /*Initialisation du pointeur courant sur le pointeur de tête*/
    Cellule_t * cour = a0;

    /*Boucle tant qu'on est pas a la fin de la liste*/
    while(cour != NULL)
    {
        /*Affichage des données*/
        fprintf(fich, "%d %d %s", cour->dd, cour->df, cour->msg);
        /*Incrémentation du pointeur courant*/
        cour = cour->ptsuiv;
    }
}
```

Ecrire:

- Principe : Fonction permettant l'écriture de la liste chainée dans le fichier dont le nom est passé en paramètre
- Entrée : Le pointeur de tête (a0) et le fichier où la liste chainée doit être affichée
- Sortie : Un code pour savoir si l'écriture c'est bien passée
- Lexique :

  Entrée :

  - a0 : adresse de la première cellule de la liste chainée
  - nomfich : variable contenant le nom du fichier

  Entrée / Sortie :

  - err : variable contenant le code de retour

  Intermédiaire :

  - fich : adresse du fichier où la liste sera affiché

```c
/* Fonction permettant l'écrire de la liste chainée dans le fichier dont le nom est passé en paramètre*/
int Ecrire(Cellule_t * a0, char * nomfich)
{
    int        err = 1; /*variable de retour si l'écriture c'est bien passé*/
    FILE     * fich; /*Initialisation du fichier*/

    /*ouverture du fichier*/
    fich = fopen(nomfich, "w");
    /*vérification si le fichier c'est bien ouvert*/
    if (fich)
    {
        /*Vérification si la liste n'est pas vide*/
        if (a0 != NULL)
        {
            /*Ecriture dans le fichier*/
            Afficher(a0, fich);
        }
        /*Fermeture du fichier*/
        fclose(fich);
    }
    else
    {
        printf("ERROR: Fichier ecriture erreur!\n");
        err = 0;
    }
    return err;
}
```

RecupererDateCour:

- Principe : Fonction permettant de récupérer la date du jour sous la forme aaaammjj
- Entrée : Pas d'entrée
- Sortie : La date du jour
- Lexique :

  Intermédiaire :

  - secondes : variable contenant le temps en secondes
  - temps : variable contenant le temps dans une structure

```c
/*Fonction permettant de récupérer la date du jour sous la forme aaaammjj */
int RecupererDateCour()
{
    time_t      secondes; /*varirable contenant le temps en secondes*/
    struct tm   temps; /*varirable contenant le temps dans une structure*/

    /*Récupération en seconde du temps écoulé depuis 1970*/
    time(&secondes);
    temps=*localtime(&secondes);
    return (1900+temps.tm_year)*10000+(temps.tm_mon+1)*100+temps.tm_mday;
}
```

AfficherControleDate:

- Principe : Procédure permettant l'affichage de la liste chaînée triée sans les dates obsolètes
- Entrée : Le pointeur de tête (a0) et le fichier où la liste chainée doit être affichée
- Sortie : Le fichier
- Lexique :

  Entrée :

  - a0 : adresse de la première cellule de la liste chainée

  Entrée / Sortie :

  - fich : adresse du fichier où la liste sera affiché

  Intermédiaire :

  - cour : adresse de l'élément courant
  - dateCourante : variable contenant la date du jour sous la forme aaaammjj

```c
/*Procédure permettant l'affichage de la liste chainée triée  sans les dates obsolètes*/
void AfficherControleDate(Cellule_t * a0, FILE* fich)
{
    Cellule_t   * cour = a0; /*Initialisation du pointeur courant sur le pointeur de tête*/
    int           dateCourante = RecupererDateCour(); /*Initialisation de la date courante*/

    /*Boucle tant qu'on n'est pas a la fin de la liste*/
    while(cour != NULL)
    {
        /*Vérification si la date est encore valide*/
        if(cour->dd > dateCourante)
        {
            /*Affichage des données*/
            fprintf(fich, "%d %d %s", cour->dd, cour->df, cour->msg);
        }
        /*Incrémentation du pointeur courant*/
        cour = cour->ptsuiv;
    }
}
```

SuppEltLCh:

- Principe : Procédure permettant la suppression d'un élément de la liste chaînée
- Entrée : L'adresse du pointeur précédent
- Sortie : L'adresse du pointeur précédent
- Lexique :

  Entrée / Sortie :

  - prec : adresse du pointeur précédent

  Intermédiaire :

  - tmp : adresse de l'élément à supprimer (variable de sauvegarde)

```c
/*Procédure permettant de supprimer un élément de la liste chainée*/
void SuppEltLCh(Cellule_t ** prec)
{
    Cellule_t * tmp; /*Variable de sauvegarde*/

    /*Sauvegarde du pointeur précédent*/
    tmp = *prec;
    /*Modification du pointeur précédent*/
    *prec = tmp->ptsuiv;
    /*Libération du message*/
    free(tmp->msg);
    /*Libération de la cellule*/
    free(tmp);
}
```

SuppDateObs:

- Principe : Procédure permettant de supprimer les éléments dans la liste chaînée dont la date de début est obsolète
- Entrée : Un pointeur sur le pointeur de tête (a0)
- Sortie : Le pointeur de tête
- Lexique :

  Entrée / Sortie :

  - a0 : adresse du pointeur de tête

  Intermédiaire :

  - prec : adresse de l'élément courant permettant le parcours de la liste chaînée
  - dateCourante : variable contenant la date du jour sous la forme aaaammjj

```c
/*Procédure permettant de supprimer les éléments dans la liste chainée dont la date de début est obsolète*/
void SuppDateObs(Cellule_t ** a0)
{
    Cellule_t  ** prec = a0;/*Initialisation du pointeur précédent (courant) sur le pointeur de tête*/
    int           dateCourante = RecupererDateCour();/*Initialisation de la date courante*/

    /*Boucle tant qu'on n'est pas à la fin de la liste et que la date n'est pas obsolète*/
    while((*prec != NULL) && ((*prec)->dd < dateCourante))
    {
        /*Supression de la cellule courante*/
        SuppEltLCh(prec);
    }
}
```

ModifierDateDébut:

- Principe : Procédure permettant de modifier la date de début des cellules ayant une certaine date de début
- Entrée : Un pointeur sur le pointeur de tête (a0) et deux entiers, la date de début à modifier et la nouvelle date
- Sortie : Le pointeur de tête
- Lexique :

  Entrée:

  - dateAModifier : variable contenant la date à modifier
  - nvDate : variable contenant la nouvelle date

  Entrée / Sortie :

  - a0 : adresse du pointeur de tête

  Intermédiaire :

  - prec : adresse de l'élément courant permettant le parcours de la liste chaînée
  - nv : adresse de la nouvelle cellule créé avec la nouvelle date

```c
/*Procédure permettant de modifier la date de début des cellules ayant une certaine date de début*/
void ModifierDateDebut(Cellule_t ** a0, int dateAModifier, int nvDate)
{
    Cellule_t   ** prec = a0;/*Initialisation du pointeur précédent (courant) sur le pointeur de tête*/
    Cellule_t    * nv = NULL;/*Déclaration de l'adresse de la nouvelle cellule*/

    /*Boucle tant qu'on n'est pas à la fin de la liste et que la date courante est inférieure ou égale à la date a modifier*/
    while((*prec != NULL) && (*prec)->dd <= dateAModifier)
    {
        /*Vérification si la date courante est égale à la date à modifier*/
        if((*prec)->dd == dateAModifier)
        {
            if(ValiditeDate(nvDate, (*prec)->df) == 0)
            {
                /*Création d'une nouvelle cellule avec la nouvelle date et */
                nv = CreerCellule(nvDate, (*prec)->df, (*prec)->msg);
                if(nv)
                {
                    SuppEltLCh(prec);
                    /*Insertion de la nouvelle cellule dans la liste chainée triée*/
                    InsererCellule(a0, nv);
                }
            }
            else
            {
                SuppEltLCh(prec);
            }
        }
        else
        {
            /*Mise à jour du pointeur précedent*/
            prec = &((*prec)->ptsuiv);
        }
    }
}
```

ValiditeDate:

- Principe : Fonction permettant de vérifier si une date de début est bien inférieure à la date de fin
- Entrée : Deux entiers, la date de début et la date de fin
- Sortie : 0 si les date sont correctes sinon 1
- Lexique :

  Entrée:

  - dd : variable contenant la date de début
  - df : variable contenant la date de fin

  Intermédiaire :

  - res : variable contenant le résultat à retourner

```c
/*Permet de vérifier si une date de début est bien inférieure à la date de fin*/
int ValiditeDate(int dd, int df)
{
    int res = 0;
    if(dd > df)
    {
        res = 1;
    }
    return res;
}
```

Libérer :

- Principe : Procédure permettant de libérer la liste chaînée à la fin du programme
- Entrée : Un pointeur sur le pointeur de tête
- Sortie : Le pointeur de tête
- Lexique :

  Entrée/Sortie :

  - a0 : adresse du pointeur de tête

  Intermédiaire :

  - cour : adresse de l'élément courant
  - tmp : adresse de l'élément à supprimer

```c
/*Permet de libérer la liste de chainée à la fin du programme */
void Liberer(Cellule_t * * a0)
{
    Cellule_t   * cour = *a0; /*Initialisation du pointeur courant sur le pointeur de tête*/
    Cellule_t   * tmp = cour; /*Initialisation du pointeur temporaire sur le pointeur courant*/

    /*Boucle tant qu'on n'est pas a la fin de la liste*/
    while (cour != NULL)
    {
        /*Incrémentation du pointeur courant*/
        cour = cour->ptsuiv;
        /*Libération du message*/
        free(tmp->msg);
        /*Libération de la cellule*/
        free(tmp);
        /*Déplacement du pointeur temporaire sur le pointeur courant*/
        tmp = cour;
    }

    *a0 = NULL;
    a0 = NULL;
    tmp = NULL;
    cour = NULL;
}
```

main.c

Dans ce fichier se trouve le programme principal.

Programme principal :

- Lexique :

  Entrée :

  - argv[1] : variable contenant le nom du fichier

  Intermédiaire :

  - codeErr : variable contenant le code à retourner à la fin du programme
  - tmpdd : variable temporaire contenant la date de début
  - tmpdf : variable temporaire contenant la date de fin
  - testEOF : variable de test de fin de fichier
  - tmpmsg : variable temporaire contenant le message
  - a0 : adresse de la première cellule de la liste chainée
  - nv : adresse de la cellule à insérer
  - fich : adresse du fichier

```c
/* ------------------------------------------------------------ */
/*                                                              */
/*  main.c   Contient le programme principal                    */
/*                                                              */
/* ------------------------------------------------------------*/

#include "tp1.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char * argv[])
{
    /*VARIABLES*/
    int           codeErr = 1;   /*Initialisation du code erreur. Renvoie 0 si tout c'est bien passée et 1 si il y a eu une erreur*/
    int           tmpdd;         /*Déclaration de la variable temporaire permettant de récupérer la date de début*/
    int           tmpdf;         /*Déclaration de la variable temporaire permettant de récupérer la date de fin*/
    int           testEOF;       /*Déclaration de la variable de test pour savoir si on est à la fin du fichier (pour l'insertion dans la liste chainée)*/
    int           dateAModifier; /*Déclaration de la variable contenant la date à modifier*/
    int           nvDate;        /*Déclaration de la variable contenant la nouvelle date*/
    char          tmpmsg[100];   /*Déclaration de la variable temporaire permettant de récupérer le message*/
    Cellule_t   * a0 = NULL;     /*Initialisation du pointeur de tête de la liste chainée*/
    Cellule_t   * nv = NULL;     /*Initialisation du pointeur du nouvelle élément*/
    FILE        * fich;          /*Initialisation du fichier*/
```

```c
/*TRAITEMENTS*/
/*Vérification qu'il y a bien le nom du fichier passé en paramètre */
if (argc == 2)
{
    /*Ouverture du fichier en lecture*/
    fich = fopen(argv[1], "r");

    /*Vérification que le fichier a bien été ouvert*/
    if (fich)
    {
        /*Récuperation du premier élément*/
        /*Récupération de la date de début et de fin*/
        testEOF = fscanf(fich, "%d %d", &tmpdd, &tmpdf);
        /*Vérification si on a bien récupérer une date de début et de fin*/
        if(testEOF != -1)
        {
            /*Récupération du message*/
            fgets(tmpmsg, 100, fich);
            tmpmsg[strlen(tmpmsg)-1]= '\n';
            /*Vérification si la date de début est bien inférieure a la date de fin*/
            if (ValiditeDate(tmpdd, tmpdf) == 0)
            {
                /*Création de la première cellule */
                a0 = CreerCellule(tmpdd, tmpdf, tmpmsg);
            }
        }
        /*Boucle tant qu'on est pas à la fin du fichier*/
        while (!feof(fich))
        {
            /*Récupération de la date de début et de fin*/
            testEOF = fscanf(fich, "%d %d", &tmpdd, &tmpdf);
            /*Vérification si on a bien récupérer une date de début et de fin*/
            if(testEOF != -1)
            {
                /*Récupération du message*/
                fgets(tmpmsg, 100, fich);
                tmpmsg[strlen(tmpmsg)-1]= '\n';
                /*Vérification si la date de début est bien inférieure a la date de fin*/
                if(ValiditeDate(tmpdd, tmpdf) == 0)
                {
                    /*Création d'une nouvelle cellule*/
                    nv = CreerCellule(tmpdd, tmpdf, tmpmsg);

                    /*Vérification qu'une cellule a bien été alloué*/
                    if(nv)
                    {
                        /*Insertion de la nouvelle cellule dans la liste chainée triée*/
                        InsererCellule(&a0, &nv);
                    }
                }
            }
        }
```

```
77
78              /*Fermeture du fichier*/
79              fclose(fich);
80              /*Ecriture dans un fichier du résultat liste chainée triée*/
81              Ecrire(a0, "sortieLChTriee.txt");
82              /*Affichage de la liste chainée triée des éléments valide*/
83              AfficherControleDate(a0, stdout);
84              /*Suppression des cellules avec des dates de début obsolètes*/
85              SuppDateObs(&a0);
86              /*Ecriture dans un fichier du résultat liste chainée triée sans les dates obsolètes*/
87              Ecrire(a0, "sortieLChDateObs.txt");
88              /*Demande de la date a modifier*/
89              printf("Entrez la date à modifier\n");
90              scanf("%d%*c", &dateAModifier);
91              /*Demande de la nouvelle date*/
92              printf("Entrez la nouvelle date\n");
93              scanf("%d%*c", &nvDate);
94              /*Modifier la date de début des cellules avec une certaine date de début*/
95              ModifierDateDebut(&a0, dateAModifier, nvDate);
96              Ecrire(a0, "sortieLChDateModifier.txt");
97
98              /*Libération de la mémoire*/
99              Liberer(&a0);
100
101             /*Modification du code erreur pour dire que tout c'est bien passé*/
102             codeErr = 0;
103         }
104         else
105         {
106             printf("Problème d'ouverture\n");
107         }
108
109     }
110
111     printf("CodeErr: %d\n", codeErr);
112     return codeErr;
113 }
114
```

Makefile

```
1   OPTIONS=-Wall -Wextra -g -ansi
2
3   TP1:tp1.o main.o
4       gcc -o TP1 tp1.o main.o
5   tp1.o:tp1.c tp1.h
6       gcc ${OPTIONS} -c tp1.c
7   main.o:main.c
8       gcc ${OPTIONS} -c main.c
9   clean:
10      rm -f tp1.o main.o *.gch
11
```

Tests

Dans cette partie nous décrivons comment nous avons réalisé nos tests.

Nous allons utiliser la commande suivante :

./TP1 fichierRemplissageListe < fichierEntreesAutomatique

Où :

- fichierRemplissageListe est le nom du fichier qui sert au remplissage de la liste chainée
- fichierEntreesAutomatique est le nom du fichier qui permet de faire les entrées utilisateur automatiquement

Fichier testProg.sh pour les tests :

```
#!/bin/sh
if [ $# -ne 1 ]
then
    echo "Il faut un fichier de remplissage"
    echo "utilisation ./testProg nomFichRemplissage"
    cdSortie=1
else
    printf '%b\n' "###Fichier de remplissage###"
    cat $1
    printf '%b\n' "\n\n###Resultat Valgrind###"
    valgrind ./TP1 $1 < parModifDate.txt
    printf '%b\n' "\n###Fichiers sortie###\n"
    printf '%b\n' "#####sortieLChTriee.txt"
    cat sortieLChTriee.txt
    printf '%b\n' "\n#####sortieLChDateObs.txt"
    cat sortieLChDateObs.txt
    printf '%b\n' "\n#####sortieLChDateModifier.txt"
    cat sortieLChDateObs.txt
    cdSortie=0
fi
exit $cdSortie
```

Cas

Dans cette partie se trouve les captures d'écrans des entrées et des sorties pour chaque cas.

Cas général

Fichier d'entrée : remplissage.txt

Commentaire :

- Quand la date de fin est inférieure à la date de début la cellule n'est pas insérée.
- Dans cet exemple, il y a une cellule sans message.

```
rhdavies@debian:~/Documents/SDD/tp1/tp1$ ./testProg.sh remplissage.txt
###Fichier de remplissage###
20190112 20201200
20190212 20191215 Je suis un message2
20190115 20201200 Je suis un message3
20190112 20190113 Je suis un message4
20190121 20201200 Je suis un message5
20180925 20190112 Je suis un message6
20180121 20201200 Je suis un message7
19970101 20190100 Je suis un message8
19970205 20201200 Je suis un message9
19981224 20180101 Je suis un message10
20190316 20201001 Je suis un message11
20201602 20201001 Je suis un message12
20190619 20201001 Je suis un message13
20201208 20201001 Je suis un message14
20190619 20201212 Je suis un message15
20201208 20201001 Je suis un message16
20190619 20201001 Je suis un message17
20190518 20201001 Je suis un message18
20201201 20201001 Je suis un message19
```

```
###Resultat Valgrind###
==3056== Memcheck, a memory error detector
==3056== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==3056== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright info
==3056== Command: ./TP1 remplissage.txt
==3056==
20190316 20201001  Je suis un message11
20190518 20201001  Je suis un message18
20190619 20201001  Je suis un message17
20190619 20201212  Je suis un message15
20190619 20201001  Je suis un message13
Entrez la date à modifier
Entrez la nouvelle date
CodeErr: 0
==3056==
==3056== HEAP SUMMARY:
==3056==     in use at exit: 0 bytes in 0 blocks
==3056==   total heap usage: 54 allocs, 54 frees, 31,196 bytes allocated
==3056==
==3056== All heap blocks were freed -- no leaks are possible
==3056==
==3056== For counts of detected and suppressed errors, rerun with: -v
==3056== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
###Fichiers sortie###

#####sortieLChTriee.txt
19970101 20190100  Je suis un message8
19970205 20201200  Je suis un message9
19981224 20180101  Je suis un message10
20180121 20201200  Je suis un message7
20180925 20190112  Je suis un message6
20190112 20190113  Je suis un message4
20190112 20201200
20190115 20201200  Je suis un message3
20190121 20201200  Je suis un message5
20190212 20191215  Je suis un message2
20190316 20201001  Je suis un message11
20190518 20201001  Je suis un message18
20190619 20201001  Je suis un message17
20190619 20201212  Je suis un message15
20190619 20201001  Je suis un message13

#####sortieLChDateObs.txt
20190316 20201001  Je suis un message11
20190518 20201001  Je suis un message18
20190619 20201001  Je suis un message17
20190619 20201212  Je suis un message15
20190619 20201001  Je suis un message13

#####sortieLChDateModifier.txt
20190316 20201001  Je suis un message11
20190518 20201001  Je suis un message18
20190619 20201001  Je suis un message17
20190619 20201212  Je suis un message15
20190619 20201001  Je suis un message13
```

Liste vide

Fichier d'entrée : remplissageVide.txt

```
rhdavies@debian:~/Documents/SDD/tp1/tp1$ ./testProg.sh remplissageVide.txt
###Fichier de remplissage###


###Resultat Valgrind###
==3063== Memcheck, a memory error detector
==3063== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==3063== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright info
==3063== Command: ./TP1 remplissageVide.txt
==3063==
Entrez la date à modifier
Entrez la nouvelle date
CodeErr: 0
==3063==
==3063== HEAP SUMMARY:
==3063==     in use at exit: 0 bytes in 0 blocks
==3063==   total heap usage: 19 allocs, 19 frees, 18,184 bytes allocated
==3063==
==3063== All heap blocks were freed -- no leaks are possible
==3063==
==3063== For counts of detected and suppressed errors, rerun with: -v
==3063== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

###Fichiers sortie###

#####sortieLChTriee.txt

#####sortieLChDateObs.txt

#####sortieLChDateModifier.txt
```

Dates de début égaux

Fichier d'entrée : remplissageDdEgaux.txt

Commentaire :

- Les éléments sont toujours insérés avant la cellule qui a la même date de début.
- Quand la date de fin est inférieure à la date de début la cellule n'est pas insérée.

```
rhdavies@debian:~/Documents/SDD/tp1/tp1$ ./testProg.sh remplissageDdEgaux.txt
###Fichier de remplissage###
20190312 20201200 Je suis un message1
20190312 20191215 Je suis un message2
20190312 20201200 Je suis un message3
20190312 20190113 Je suis un message4
20190312 20201200 Je suis un message5
20190312 20190112 Je suis un message6
20190312 20201200 Je suis un message7
20190312 20190100 Je suis un message8
20190312 20201200 Je suis un message9
20190312 20180101 Je suis un message10


###Resultat Valgrind###
==3072== Memcheck, a memory error detector
==3072== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==3072== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright info
==3072== Command: ./TP1 remplissageDdEgaux.txt
==3072==
20190312 20201200  Je suis un message9
20190312 20201200  Je suis un message7
20190312 20201200  Je suis un message5
20190312 20201200  Je suis un message3
20190312 20191215  Je suis un message2
20190312 20201200  Je suis un message1
Entrez la date à modifier
Entrez la nouvelle date
CodeErr: 0
==3072==
==3072== HEAP SUMMARY:
==3072==     in use at exit: 0 bytes in 0 blocks
==3072==   total heap usage: 34 allocs, 34 frees, 30,748 bytes allocated
==3072==
==3072== All heap blocks were freed -- no leaks are possible
==3072==
==3072== For counts of detected and suppressed errors, rerun with: -v
==3072== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
###Fichiers sortie###

#####sortieLChTriee.txt
20190312 20201200  Je suis un message9
20190312 20201200  Je suis un message7
20190312 20201200  Je suis un message5
20190312 20201200  Je suis un message3
20190312 20191215  Je suis un message2
20190312 20201200  Je suis un message1

#####sortieLChDateObs.txt
20190312 20201200  Je suis un message9
20190312 20201200  Je suis un message7
20190312 20201200  Je suis un message5
20190312 20201200  Je suis un message3
20190312 20191215  Je suis un message2
20190312 20201200  Je suis un message1

#####sortieLChDateModifier.txt
20190312 20201200  Je suis un message9
20190312 20201200  Je suis un message7
20190312 20201200  Je suis un message5
20190312 20201200  Je suis un message3
20190312 20191215  Je suis un message2
20190312 20201200  Je suis un message1
```

Date de fin inférieure date début

Fichier d'entrée : remplissageDfinDd.txt

Commentaire :

- Quand la date de fin est inférieure à la date de début la cellule n'est pas insérée.

```
rhdavies@debian:~/Documents/SDD/tp1/tp1$ ./testProg.sh remplissageDfInfDd.txt
###Fichier de remplissage###
21190412 20201200 Je suis un message1
21200412 21200608 Je suis un message2
21190512 20201200 Je suis un message3


###Resultat Valgrind###
==3093== Memcheck, a memory error detector
==3093== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==3093== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright info
==3093== Command: ./TP1 remplissageDfInfDd.txt
==3093==
21200412 21200608  Je suis un message2
Entrez la date à modifier
Entrez la nouvelle date
CodeErr: 0
==3093==
==3093== HEAP SUMMARY:
==3093==     in use at exit: 0 bytes in 0 blocks
==3093==   total heap usage: 24 allocs, 24 frees, 30,518 bytes allocated
==3093==
==3093== All heap blocks were freed -- no leaks are possible
==3093==
==3093== For counts of detected and suppressed errors, rerun with: -v
==3093== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

###Fichiers sortie###

#####sortieLChTriee.txt
21200412 21200608  Je suis un message2

#####sortieLChDateObs.txt
21200412 21200608  Je suis un message2

#####sortieLChDateModifier.txt
21200412 21200608  Je suis un message2
```

Message supérieur 100

Fichier d'entrée : remplissageMsgSup100.txt

Commentaire :

- Si le message dépasse 100 caractères, nous avons décidé de séparer le message dans différentes cellules. Ces cellules auront la même date de début et de fin.

```
rhdavies@debian:~/Documents/SDD/tp1/tp1$ ./testProg.sh remplissageMsgSup100.txt
###Fichier de remplissage###
20190112 20301200 Je suis un message1
20200114 20301200 Je suis un message qui est extrement long afin de verifier ce qu'il se passe. Ce messag
e n'est toujours pas assez long... Parceque 4.
20210112 20301200 Je suis un message2
20220112 20301200 Je suis un message3


###Resultat Valgrind###
==3103== Memcheck, a memory error detector
==3103== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==3103== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright info
==3103== Command: ./TP1 remplissageMsgSup100.txt
==3103==
20200114 20301200 jours pas assez long... Parceque 4.
20200114 20301200  Je suis un message qui est extrement long afin de verifier ce qu'il se passe. Ce messa
ge n'est to
20210112 20301200  Je suis un message2
20220112 20301200  Je suis un message3
Entrez la date à modifier
Entrez la nouvelle date
CodeErr: 0
==3103==
==3103== HEAP SUMMARY:
==3103==     in use at exit: 0 bytes in 0 blocks
==3103==   total heap usage: 32 allocs, 32 frees, 30,795 bytes allocated
==3103==
==3103== All heap blocks were freed -- no leaks are possible
==3103==
==3103== For counts of detected and suppressed errors, rerun with: -v
==3103== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
###Fichiers sortie###

#####sortieLChTriee.txt
20190112 20301200  Je suis un message1
20200114 20301200 jours pas assez long... Parceque 4.
20200114 20301200  Je suis un message qui est extrement long afin de verifier ce qu'il se passe. Ce messa
ge n'est to
20210112 20301200  Je suis un message2
20220112 20301200  Je suis un message3

#####sortieLChDateObs.txt
20200114 20301200 jours pas assez long... Parceque 4.
20200114 20301200  Je suis un message qui est extrement long afin de verifier ce qu'il se passe. Ce messa
ge n'est to
20210112 20301200  Je suis un message2
20220112 20301200  Je suis un message3

#####sortieLChDateModifier.txt
20200114 20301200 jours pas assez long... Parceque 4.
20200114 20301200  Je suis un message qui est extrement long afin de verifier ce qu'il se passe. Ce messa
ge n'est to
20210112 20301200  Je suis un message2
20220112 20301200  Je suis un message3
```