Chapitre 9 : liste linéaire chaînée (suite)

Manipulation d'une liste linéaire chaînée :

1) Affichage le contenu d'une liste linéaire chaînée :

Pour parcourir un tableau on utilise les indices :

```
for ( i = 0 ; i < nbElem ; i++) .....
```

Pour parcourir un fichier, comme on n'a pas d'indices, on utilise la boucle while :

```
while (!feof(...)) ....
```

Pour parcourir une liste linéaire chaînée :

1. on utilise la boucle :

```
while (liste != NULL) .....
```

ou plus court :

```
while (liste) .....
```

2. on avance dans la liste pour chaque tour de boucle :

```
liste = liste->suivant ; /* liste pointe comme son "suivant" */
```

Exemple 1 :

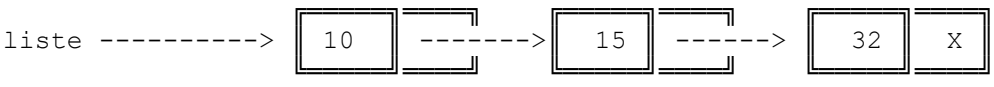
Écrire une fonction permettant d'afficher le contenu d'une liste linéaire chaînée des entiers.

Solution :

```
void afficher( Pointeur liste)
{
  while (liste) /* Tantque la liste soit non vide */

     { printf("%5d\n", liste -> valeur );

       liste = liste -> suivant ; /* avancer dans la liste */
     }
}
```

Exercice :

Simuler la fonction avec la liste suivante :



Exemple 2 :

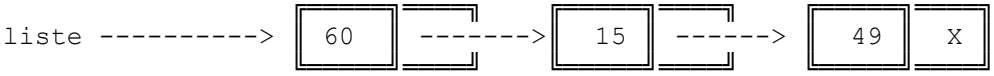
Écrire une fonction "récursive" permettant d'afficher le
contenu d'une liste linéaire chaînée des entiers.

Solution :

```
void afficher( Pointeur liste)
{
  if (liste)
     { printf("%5d\n", liste -> valeur );
       afficher(liste -> suivant) ;
     }
}
```

Exercice :

Simuler la fonction avec la liste suivante :



2) Recherche d'un élément dans une liste linéaire chaînée :

2.1) Raisons d'une recherche :

On cherche un élément de la liste :

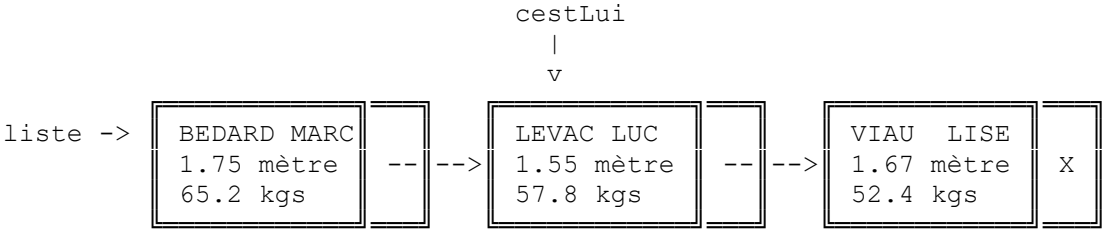
1. pour une consultation (cas très fréquent)
2. pour une suppresion (exemple : cas d'abandon dans un système d'inscription)
3. pour une modification (exemple : changements de notes dans un système de gestion des notes)

etc ....

Pour les besoins de consultation, il suffit d'avoir un pointeur qui :

. pointe vers rien si on ne trouve pas OU
. pointe vers l'élément trouvé

Si on cherche "LEVAC LUC" et si on dispose d'un pointeur "cestLui" :



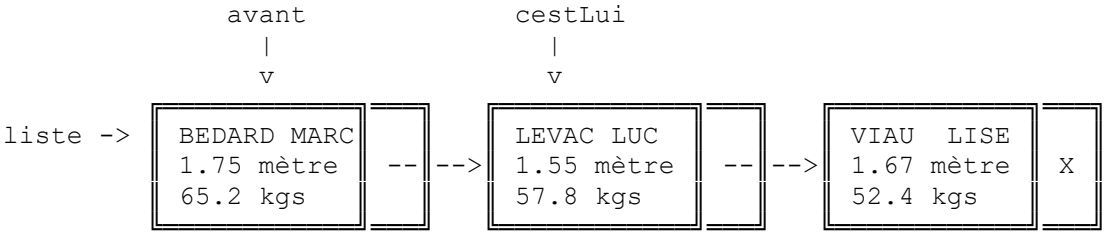
on est capable de faire : afficher(cestLui->pers) ; où pers est
le premier champ du type Element.

Si on a besoin de corriger la taille de "LEVAC LUC" : 175 mètre
au lieu de 1.55 mètre, il suffit de faire :

```
cestLui->pers.taille = 1.75 ;
```

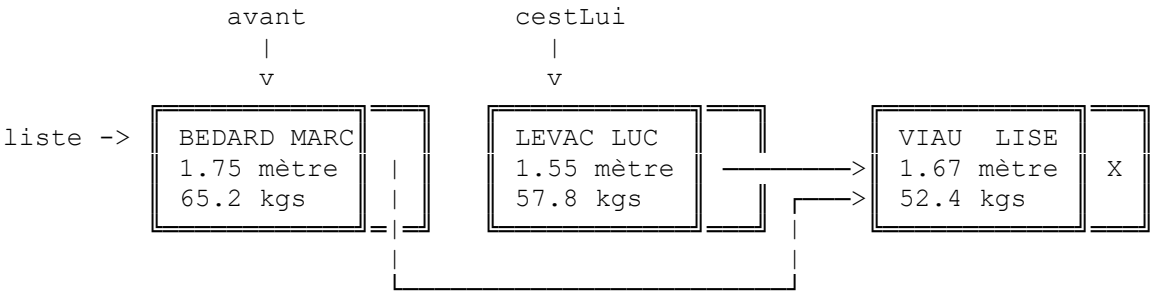
Cependant, si on a besoin de supprimer "LEVAC LUC" de la liste,
le pointeur cestLui n'est pas suffisant.

Supposons qu'on dispose d'un autre pointeur du nom "avant" qui
pointe vers le prédécesseur de l'élément trouvé :



Pour supprimer "LEVAC LUC", il suffit de faire comme suit :

```
avant->suivant = cestLui->suivant ;
```



Résumés :

Ainsi, pour les besoins de la gestion d'une liste (y compris
la suppression), on aimerait avoir une fonction qui retourne
2 pointeurs :

1. cestLui qui pointe sur l'élément trouvé ou vaut NULL
   dans le cas contraire

2. avant qui pointe sur le prédécesseur de l'élément trouvé
   ou qui vaut NULL si on trouve au début de la liste.

2.2) Recherche dans une liste triée des personnes :

Je présente ici un cas où la clé est une chaîne de caractères.
C'est plus simple si la clé est un entier :

```
void chercher ( char * aChercher, Pointeur liste,
                      Pointeur *av, Pointeur * cl)

 {  int longueur = strlen(aChercher), compare , trouve ;
    Pointeur avant, cestLui          ; /* comme *av et *cl */

    /* Initialisation : */

    avant    = NULL   ;
    cestLui = liste  ;
    trouve   = 0      ; /* Faux, on ne trouve pas encore */

    while (cestLui && !trouve)
      {
       compare = strnicmp (cestLui->pers.nomPre, aChercher,
                                        longueur);

       trouve  = compare == 0 ; /* on trouve */

       if (!trouve )
        if ( compare < 0) /* on a la chance et on continue à chercher */
             { avant = cestLui ;
              cestLui = cestLui->suivant;
             }
        else    cestLui = NULL ; /* on ne trouve plus */
      }

    *av = avant ;
    *cl = cestLui ;
 }
```

Notes :

1. La fonction :

   strnicmp(chaine1, chaine2, k)

   permet de comparer k premiers caractères de ces deux chaînes sans tenir compte des majuscules ou des minuscules. Le résultat (variable compare dans la fonction) est un entier :

   ```
   zéro  : elles sont pareilles dans les k premiers caractères
    < 0  : chaine1 < chaine2 dans les k premiers caractères
    > 0  : chaine1 > chaine2 dans les k premiers caractères
   ```

2. Pour chaque tour de la boucle de recherche :

   Si compare vaut zéro, on trouve. C'est réglé.

   Si compare < 0, on ne trouve pas encore mais, on a quand même la chance car le nom de la personne rendue (exemple "BEDARD MARC") est plus petit que le nom recherché (ici "LEVAC LUC"). Dans ce cas, avant prend la position de cestLui et on avance cestLui.

Si on cherche "LAVOIE JACQUES" et qu'on trouve plutôt
"LEVAC LUC", dans ce cas on a compare > 0 et on n'aura
jamais la chance de trouver "LAVOIE JACQUES" car la liste
est triée. On affecte NULL à cestLui pour signaler qu'on
ne rencontre pas "LAVOIE JACQUES".

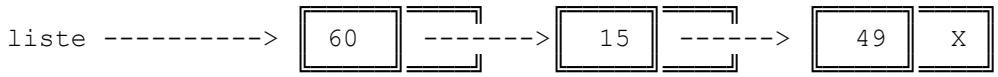
3. Notez que :

   a) la recherche de "BEDARD MARC" donne NULL au pointeur
      avant. Dans ce cas, cestLui qui est non NULL alors
      que avant vaut NULL signale qu'on trouve au début
      de la liste. Cette information est précieuse
      pour la suppression.

   b) la recherche de "LAVOIE JACQUES" est échouée. cestLui
      vaut NULL pour signaler qu'on ne le trouve pas. Cependant
      avant pointe sur "BEDARD MARC" (le prédécesseur immédiat
      si "LAVOIE JACQUES" est dans la liste). Cette information
      est précieuse pour l'ajout de "LAVOIE JACQUES" dans la liste.

2.3) Recherche dans une liste non triée :

Je présente ici un cas où la clé est un entier.




```
void chercher(int aChercher, Pointeur liste, Pointeur * av,
                        Pointeur * cl)
{
    Pointeur avant = NULL, cestLui = liste ;

    /* Si on ne trouve pas, on avance dans la liste */
    while (cestLui && cestLui->valeur != aChercher)
       { avant = cestLui ;
         cestLui = cestLui->suivant ;
       }

    *av = avant ;
    *cl = cestLui ;
}
```

Exercice 1 :

Simuler l'algorithme avec la recherche de 15 et de 20 de la
liste des entiers représentés par le schéma ci-dessus.

Exercice 2 :

Réécrire la fonction de recherche dans le cas où la clé est
un nom et un prénom et, qu'il il y a possibilité de noms doubles :

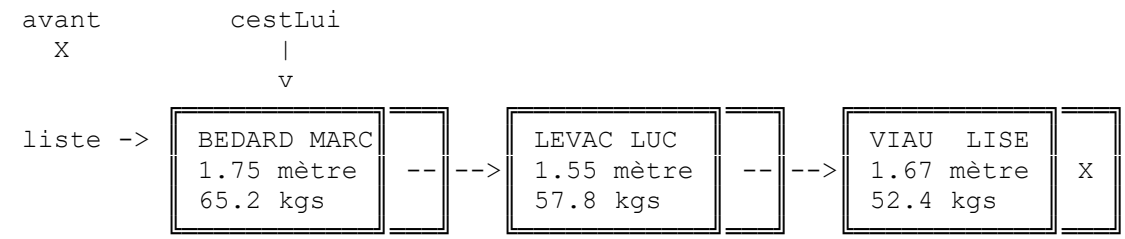
1. si la liste est triée selon le nom et prénom
2. si la liste n'est pas du tout triée.

3) Suppression d'un élément dans une liste linéaire chaînée :

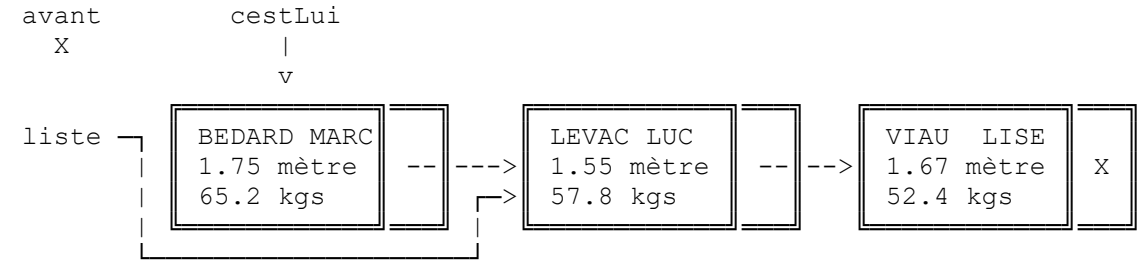
Cas 1 : élément à supprimer est le premier élément de la liste

Dans ce cas : cestLui != NULL et avant == NULL

Après la recherche, avant la suppression:



Après la suppression:

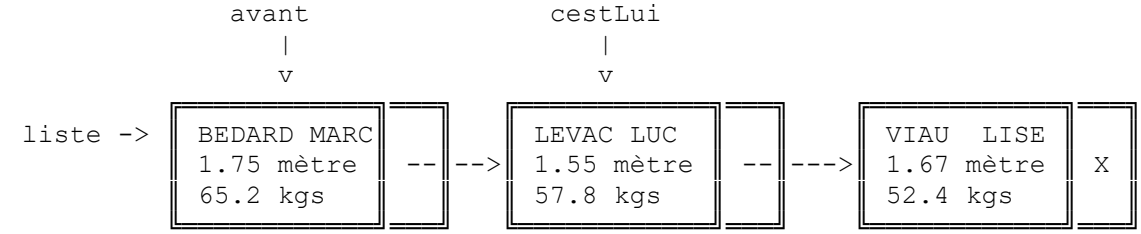


On a fait :

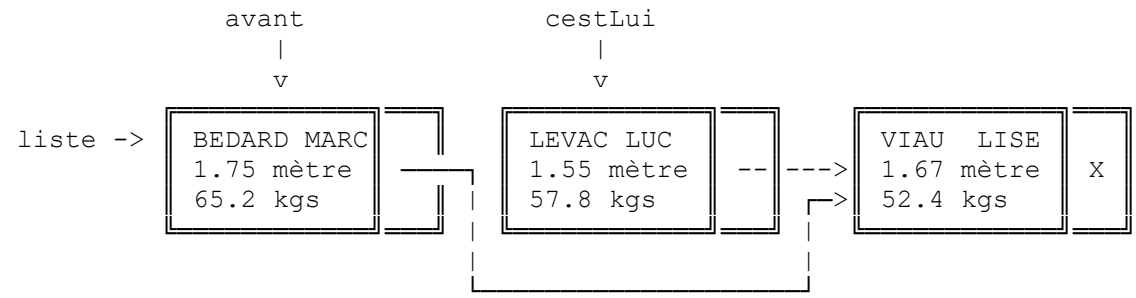
```
if (avant == NULL)
      liste = liste->suivant ;
else
       .... à venir ....
```

Cas 2 : élément à supprimer n'est pas le premier élément de la liste

Après la recherche, avant la suppression:



Après la suppression:



On a programmé :

```
        else
                avant->suivant = cestLui->suivant;
```

Fonction de suppression :

Appel (utilisation) :

```
     chercher(aChercher, liste, &avant, &cestLui);
     if (cestLui) /* cas trouvé */
       {
           afficher(cestLui->per);

           printf("Confirmez-vous la suppression ? (O/N) ");
           fflush(stdin);
           if (toupper(getchar()) == 'O')

                supprimer(&liste, avant, cestLui);
       }
     ......
```

La fonction :

```
   void supprimer (Pointeur * P, Pointeur avant, Pointeur cestLui)

    {
        if (avant == NULL) /* trouvé au début de la liste */

            *P = (*P)->suivant ;

        else
               avant->suivant = cestLui->suivant;

        free(cestLui);

    }
```

4) Ajout d'un nouvel élément :

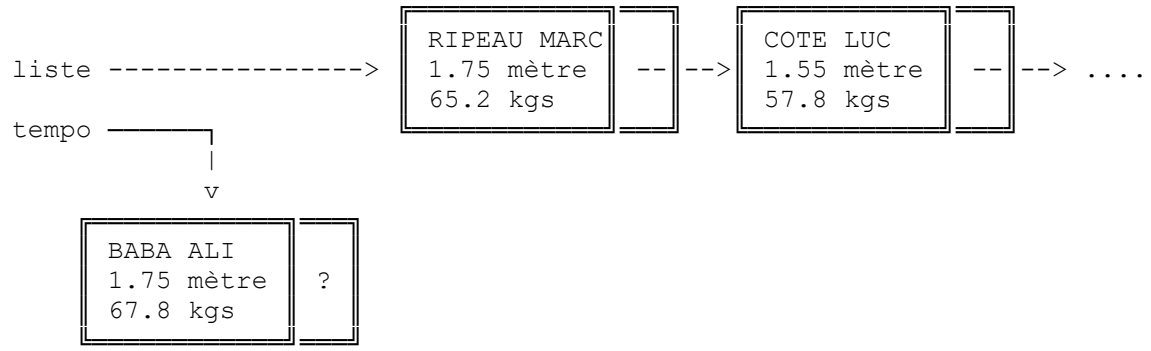
Cas 1 : la liste n'est pas triée :

Dans un tel cas, il suffit d'ajouter, par exemple, au début de la liste (le cas d'une pile (d'assiettes!)) :

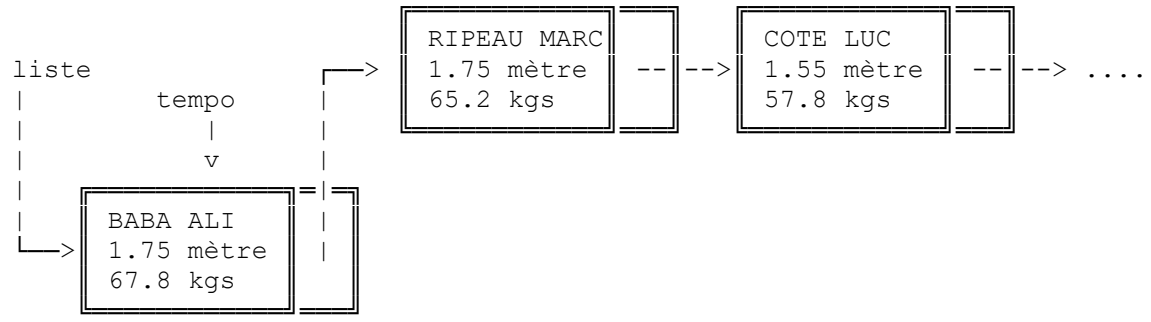
avec ces 3 instructions :

```
obtenir(&nouPers) ; /* informations d'une nouvelle personne */

tempo = (Element *) malloc (sizeof(Element));

tempo->pers = nouPers ;
```

on a ce qui suit :



Ce qu'on veut, finalement, est :



Autrement dit :

```
tempo->suivant = liste ;
liste          = tempo ;
```

Exercice :

Peut-on écrire comme suit ? :

```
liste = tempo ;
tempo->suivant = liste ;
```

Justifiez votre réponse.

Exercice :

Après avoir ajouté le nouvel élément, peut-on faire : free (tempo); ?
Justifiez votre réponse.

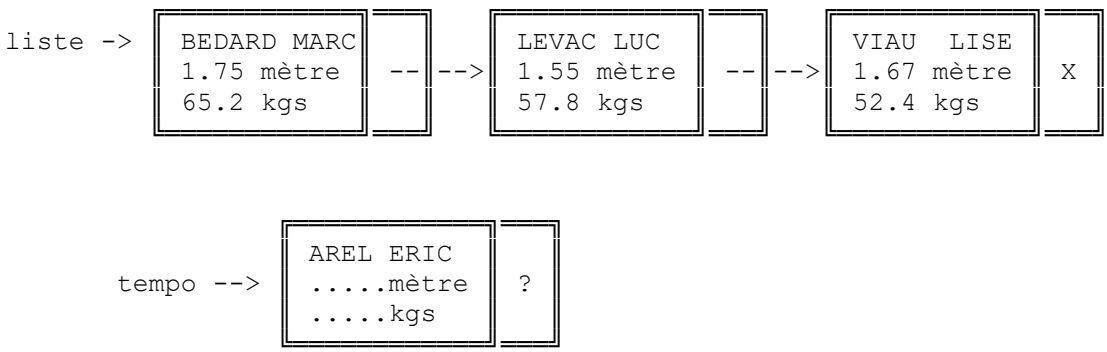
Cas 2 : la liste est triée :

Dans ce cas, on se préoccupe de l'ajouter à la bonne place
(c'est-à-dire que la liste sera encore triée après l'ajout).

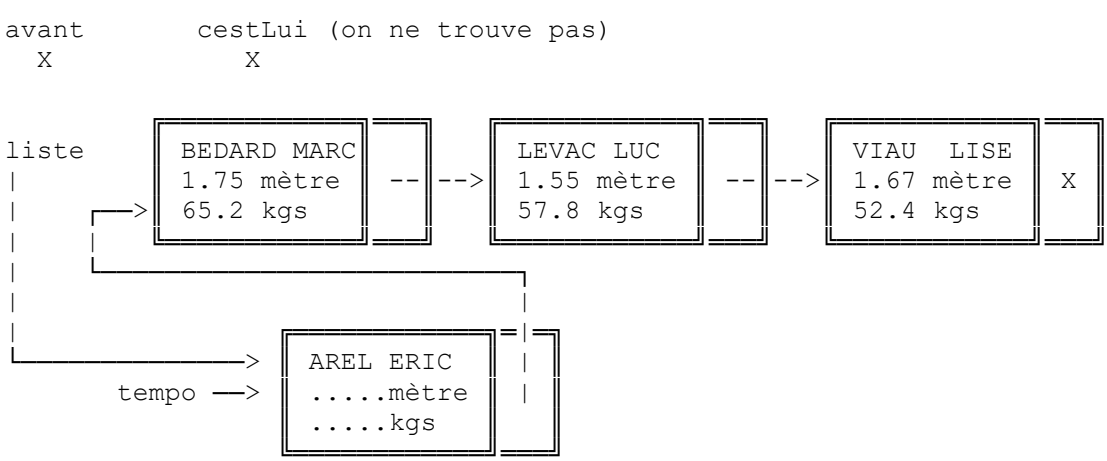
```
if (avant != NULL)
  { tempo->suivant = avant->suivant ;
    avant->suivant = tempo ;
  }
```

Cependant si on veut ajouter "AREL ERIC" dans la liste, on
aimerait qu'il soit placé au début de la liste.

Revenons avec la recherche de cette personne :



Ce qu'on veut est :



Il faut programmer comme suit :

```
       .....
       if (avant == NULL)
        {  tempo->suivant = liste ;
           liste          = tempo ;
        }
```

Résumé de la programmation :

```
    ......
    obtenir(&nouPers);

    chercher(nouPers.nomPre, liste, &avant, &cestLui);

    if (cestLui == NULL) Ajouter (nouPers, &liste, avant);
```

La fonction ajouter :

```
      void ajouter (personne nouPers, Pointeur * P, Pointeur avant)

      {  Pointeur tempo , liste = *P ;

         tempo = (Element *) malloc (sizeof(Element));

         if (avant == NULL) /* ajouter au début de la liste */

              { tempo->suivant = liste ;
                liste          = tempo ;
              }

          else /* ajouter après avant */

              { tempo->suivant = avant->suivant ;
                avant->suivant = tempo ;
              }

         *P = liste ;
      }
```

5) Modification des informations d'un élément :

Cas 1 :

La liste n'est pas triée ou la clé n'est pas modifiée.

```
chercher (cle, liste, &avant, &cestLui);

demander(&unePers); // nouvelles informations

cestLui->pers = unePers ;
```

Cas 2 :

La liste est triée et la clé est modifiée :

```
chercher (cle, liste, &avant, &cestLui);

/* Supprimer */

supprimer(&liste, avant, cestLui);

demander(&unePers);

ajouter (unePers, &liste, avant);
```

6) Calcul des statistiques :

Exemple 1 :

Écrire une fonction et deux appels de cette fonction pour afficher à l'écran :
- la taille moyenne
- le poids moyen

des personnes d'une liste linéaire des personnes.

Solution :

```
float moyenne ( Pointeur liste, char codeVoulu )
{ float somme = 0.0 ;
  int   n     =  0  ;

  while (liste)
  { n++ ;
    if (codeVoulu == 'T')
           somme += liste->pers.taille ;
    else
           somme += liste->pers.poids  ;

    liste = liste->suivant ;
  }
  return somme / n ;
}
```

Appels :

```
printf("La taille moyenne : %6.2f mètre\n", moyenne(liste, 'T'));
printf("Le poids  moyen   : %6.2f kgs\n", moyenne(liste, 'P'));
```

Exercice 1 :

Écrire une fonction et deux appels de cette fonction pour afficher à l'écran :
- la taille la plus grande et la taille la plus petite
- le poids le plus lourd et le poids le plus léger

7) Autres gestions d'une liste linéaire :

Exemple 1 : création d'un fichier binaire :

Écrire une fonction permettant de créer un fichier binaire des personnes à partir d'une liste linéaire des personnes.

Solution :

```
void creer (char * nomACreer, Pointeur liste)
{
   FILE * aCreer = fopen(nomACreer, "wb");

   int nbOctets = sizeof(personne);

   while (liste)
     {
        fwrite(&(liste->pers), nbOctets, 1, aCreer);
        liste = liste->suivant;
     }
   fclose(aCreer);
}
```

Appel :

```
Creer ("personne.Bin", liste);
```

Exercice 1 :

Écrire une fonction et 2 appels de cette fonction pour :

- créer un fichier binaire "GRANDE.PER" des personnes dont la taille est supérieur à 1.85 mètre ;
- créer un fichier binaire "GROS.PER" des personnes dont le poids est supérieur à 123.4 kgs ;

Exercice 2 :

Écrire un programme permettant de gérer une pile d'entiers. Les actions prévues sont :

- tester si une pile est vide ou non
- fournir la valeur au sommet de la pile
- ajouter un nouvel entier au sommet de la pile
- obtenir l'élément au sommet de la pile

Exercice 3 (un défi) :

Programmer le Quick Sort sans utiliser la récursivité.

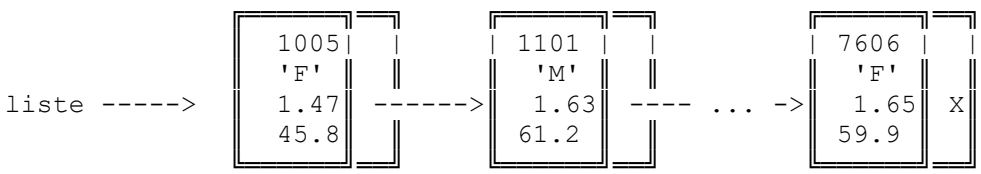
8) Exemple de révision de liste linéaire chaînée :

On dispose du fichier des données "Metrique.xxx" qui contient les informations d'une personne par ligne : numéro, sexe, taille et poids
Le fichier a été trié selon le numéro.

Exemple de données :

```
1005   1.47  45.81  F
1101   1.63  61.23  M
1103   1.70  64.86  M
1104   1.75  76.20  M
...
7605   1.78  75.75  M
7606   1.65  59.87  F
```

Écrire un programme qui permet de créer une une liste chaînée dans l'ordre FIFO des personnes :



Le programme offre aussi un menu avec des options de gestion d'une liste linéaire (recherche, suppression, ajout, modification, etc...) et traite le choix de l'usager jusqu'à ce que l'usager décide de quitter.

Solution :

```
/*  LISTE LINEAIRE CHAINEE : Fichier : LISTE.A95   */

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

/* déclarations globales */
/* déclarer le type personne */

typedef struct
         { int    numero ;
           char   sexe   ;
           float  taille, poids ;
         } personne ;

/* déclarer le type Element pour chaque élément de la liste */
typedef struct elem
         { personne pers         ;
           struct elem * suivant ;
         }
        Element;

/* déclarer le type pointeur vers un élément de la liste */
typedef Element * pointeur ;

pointeur liste;

void obtenir (char * pt)

/* Obtenir le choix de l'usager pour une option du menu */

{ char c ;

  printf("Tapez : \n");
  printf("  - Q pour Quitter \n");
  printf("  - V pour Visualiser \n");
  printf("  - C pour Chercher\n");
  printf("  - A pour Ajouter une nouvelle personne\n");
  printf("  - M pour Modifier les informations d'une personne\n");
  printf("  - S pour Supprimer une personne de la liste\n");
  printf("  - I pour visualiser la liste Inverse \n");

  do
  {
     printf("\n\n");
     printf("Entrez votre choix parmi Q, V, C, A, M, S ou I >> ");
     fflush(stdin);
     c = toupper(getchar());
  } while ( strchr("QVCAMSI", c) == NULL );

 /* équivalent à : while ( c != 'Q' && c != 'V' && c != 'C' && c != 'A'
                   && c != 'M' && c != 'S'  && c != 'I' ); */
    *pt = c ;
}
```

```
void afficher (personne unePers)

{
  printf("\n\nNuméro      : %6d\n", unePers.numero);
  printf("taille      : %6.2f\n", unePers.taille);
  printf("poids       : %6.1f\n", unePers.poids );
  printf("sexe        : %s\n", toupper(unePers.sexe) == 'F' ?
                          "Féminin" : "Masculin") ;
}


void demander ( personne * ptr )
{ personne unePers;
      printf("Entrez les informations de la personne : ");
      printf("  - Numéro : ");
      scanf("%d", &(unePers.numero));
      flushall();
      printf("  - sexe   : ");
      unePers.sexe = toupper(getche());
      printf("  - taille : ");
      scanf("%f", &(unePers.taille));
      printf("  - poids  : ");
      scanf("%f", &(unePers.poids)) ;

      *ptr = unePers;
}

void lire (FILE * donnees, personne * P)
/* lire une ligne du fichier */
{char   jeter;
 float  taille, poids ;
 fscanf(donnees,"%d%f%f%c%c%c\n", &(P->numero), &taille, &poids,
            &jeter, &jeter,&(P->sexe));
 P->taille = taille ;
 P->poids  = poids  ;
}

void visualiser ( pointeur liste)
{  const int parEcran = 20 ; /* 20 personnes par écran */
   int n ;
   char unCar ; /* pour quitter quand l'usager le désire */

   n = 0 ;
   unCar = ' ' ; /* différent à 'T' pour faire démarrer la boucle */
   printf("Dans l'ordre FIFO: \n");
   while ( unCar != 'T' && liste )
     {  n++;
        printf("%3d) %5d %6.2f %6.2f%3c\n", n, (liste->pers).numero,
         (liste->pers).taille, (liste->pers).poids,(liste->pers).sexe);
        liste = liste->suivant;

        if ( n % parEcran == 0 || liste == NULL)
        { printf("\n\n");
          printf("Appuyez sur Entrée pour continuer ou T pour terminer ");
          fflush(stdin);
          unCar = toupper(getchar());
```

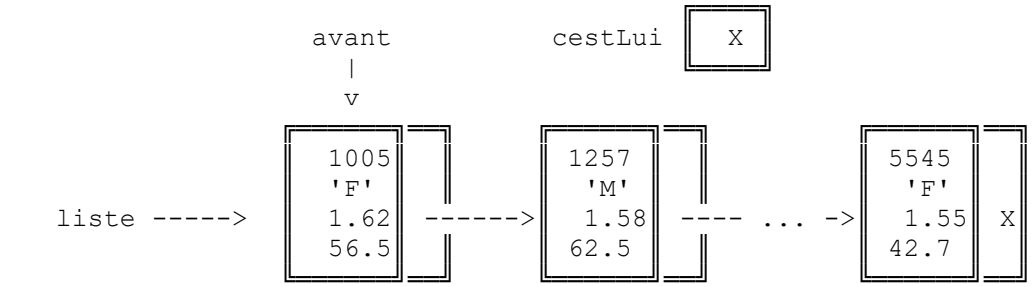
```
          if (  unCar != 'T' && liste != NULL)
           {
                printf("Dans l'ordre FIFO: \n");
           }
        }
     }
}

void creerFIFO (pointeur * L) /* dans l'ordre FIFO */
/* comme le fichier de données a été trié selon le numéro de personnes,
   cette fa‡on de création permet d'obtenir une liste triée.
*/
{
  FILE * donnees ;
  personne unePers ;
  pointeur tempo, Laliste, present ;
  char jeter;
  int tailleOctets = sizeof(Element) ;

  donnees = fopen("Metrique.xxx","r");

  if (feof(Donnees))
       Laliste = NULL ;
  else
  {   /* voici une version permettant la création dans l'ordre FIFO
         qui fait pointer d'abord la tête de liste au premier élément
         et fait allonger la liste après. On évite ainsi :
           - le Warning (avertissement) du TURBO C
           - le test if (liste == NULL) dans une boucle
      */

      Laliste = (pointeur) malloc (tailleOctets);
      lire(donnees, &unePers);
      laListe->pers = unePers;
      present = laListe;

      while (!feof(donnees))
      {
        tempo = (pointeur) malloc (tailleOctets);
        lire(donnees, &unePers);

        tempo->pers    = unePers ;
        present->suivant = tempo;
        present = tempo;
       }
      fclose(donnees);
      present->suivant = NULL; /* mettre la fin de la liste, ici
                                  en dehors de la boucle */
  }
   *L = laListe;
}
```

```
void inserer (pointeur * P, personne unePers, pointeur avant)
{ pointeur tempo;

         tempo = (pointeur) malloc(sizeof(Element));
         tempo->pers = unePers ;
         if    (avant == NULL)
           { tempo->suivant = *P ;
             (*P) = tempo;
           }
         else
           { tempo->suivant = avant->suivant ;
             avant->suivant= tempo;
           }
}

void find (int aChercher, pointeur liste, pointeur * av, pointeur * cl)
/* donnez la valeur aChercher, je cherche dans la liste et vous
   retourne deux pointeurs :
     1) *cl qui pointe vers rien si on ne le trouve pas
                    vers l'élément on cherche si on le trouve

     2) *av     qui pointe vers élément ayant la valeur qui précède
              la valeur cherchée. Ce pointeur est utile pour les
              tâches de supprimer, ajouter, modifier

   Exemple : si on cherche la personne ayant le numéro 1257, on aura :
                  avant          cestLui
                    |              |
                    |              |
                    v              v
                ┌──────┐┌──┐    ┌──────┐┌──┐         ┌──────┐┌──┐
                │ 1005 ││  │    │ 1257 ││  │         │ 5545 ││  │
                │ 'F'  ││  │    │  'M' ││  │         │  'F' ││  │
  liste -----> │ 1.62 ││  │ -----> 1.58 ││  │ ---- ... -> 1.55 ││ X│
                │ 56.5 ││  │    │ 62.5 ││  │         │ 42.7 ││  │
                └──────┘└──┘    └──────┘└──┘         └──────┘└──┘
*/
{  pointeur avant, cestLui ;
   avant = NULL ;
   cestLui = liste ;
   while ( cestLui && (cestLui->pers).numero < aChercher )
    {  avant   = cestLui;
       cestLui = cestLui->suivant ;
    }
   /* si on ne le trouve pas, on met le cestLui à NULL */
   if (cestLui && (cestLui->pers).numero != aChercher)
       cestLui = NULL;
   *av = avant ;
   *cl = cestLui ;
}
```

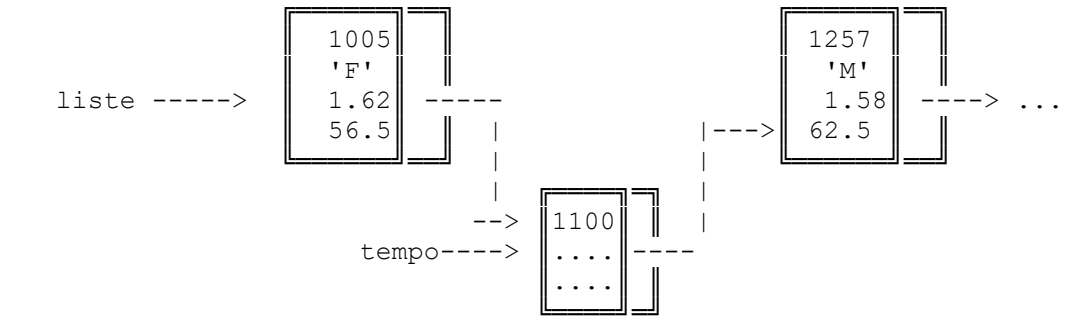
```
void inverser ( pointeur * L )
{ /* A simuler pour mieux comprendre le principe */

  pointeur aContourner, leSuivant , liste = * L ;

  aContourner = liste ;
  liste        = NULL  ;

  while (aContourner)
   { leSuivant            = aContourner->suivant ;

     aContourner->suivant = liste ;
     liste                = aContourner ;

     aContourner          = leSuivant;
    }

  *L = liste ;
}

void chercher (pointeur liste)
{  pointeur  cestLui, avant ;
   int aChercher;
   do
   {
     printf("Quelques numéros existants   : 1005, 1101, 1707, 7104, 7506\n");
     printf("Quelques numéros inexistants : 1899, 2222, 5678\n");
     printf("Entrez le numéro de la personne à chercher ");
     scanf("%d", &aChercher);
     find (aChercher,liste, &avant, &cestLui);

      if (cestLui != NULL) /* on le trouve */
       {
           printf("\nYOUPI! on l'a trouvé : \n");
           afficher(cestLui -> pers);
       }
      else
       printf("Désolé! on ne trouve pas cette personne\n");
      /* exemple "pédagogique" */
        if (avant == NULL)
           printf("avant vaut NULL\n");
        else
           printf("valeur avant : %5d\n", avant->pers.numero);

      printf("avez-vous une autre personne à chercher (O/N) ? ");
      fflush(stdin);
     }
  while (toupper(getchar()) != 'N');
}

void eliminer ( pointeur * P, pointeur avant, pointeur cestLui)
{         if    (avant == NULL) /* au début de la liste */
              *P = (*P)->suivant ; /* on fait avancer la liste */
         else  avant->suivant = cestLui->suivant;
}
```

```
void supprimer (pointeur * P)
/* Si on veut supprimer 1257 :
     a) avec find ....
                  avant          cestLui
                    |              |
                    v              v
                 ┌──────┐       ┌──────┐          ┌──────┐
                 │ 1005 │       │ 1257 │          │ 5545 │
                 │ 'F'  │       │ 'M'  │          │ 'F'  │
  liste -----> ──│ 1.62 │──────>│ 1.58 │── ... ->─│ 1.55 │ X
                 │ 56.5 │       │ 62.5 │          │ 42.7 │
                 └──────┘       └──────┘          └──────┘
     a) On supprime :
                  avant          cestLui
                    v              v
                 ┌──────┐       ┌──────┐          ┌──────┐
                 │ 1005 │       │ 1257 │          │ 1265 │
                 │ 'F'  │       │ 'M'  │          │ 'M'  │
  liste -----> ──│ 1.62 │──┐    │ 1.58 │─────────>│ 1.75 │──> ....
                 │ 56.5 │  |    │ 62.5 │    -->   │ 92.7 │
                 └──────┘  |    └──────┘          └──────┘
                           |-----------------|
*/

{  pointeur  cestLui, avant, tempo ;
   int aSupprimer;
   char reponse ;
   do
   {
    printf("Quelques numéros existants   : 1005, 1101, 1707, 7104, 7506 \n");
    printf("Quelques numéros inexistants : 1899, 2222, 5678\n");
    printf("\n\nEntrez le numéro à supprimer ");
    scanf("%d", &aSupprimer);
    find (aSupprimer,liste, &avant, &cestLui);
    reponse = 'N';
    if (cestLui != NULL)
    {
       printf("On l'a trouvé : \n");
       afficher(cestLui->pers);
       printf("Confirmez-vous la suppression ?(O/N) ");
       fflush(stdin);
       reponse = toupper(getche());

       if (reponse == 'O') eliminer ( P, avant, cestLui);
       }
      else
       printf("Désolé! on ne trouve pas cette personne\n");

      printf("avez-vous autre personne à supprimer (O/N) ? ");
      fflush(stdin);

     }
  while (toupper(getche()) != 'N');
}
```

```
void ajouter (pointeur * P)
/*  Supposons qu'on veuille ajouter une personne de numéro 1100
    a) avec find .... on ne le trouve pas mais avant pointe sur
       l'élément ayant le numéro 1005
```




```
    b) on ajoute :
```




```
*/

 {  pointeur cestLui, avant, tempo ;
    int aAjouter;
    char reponse ;
    personne unePers ;
    float x, y;
       clrscr();
       Demander(&unePers);
       Find (unePers.numero,liste, &avant, &cestLui);
       reponse = 'N';
       if (cestLui != NULL)
        {
        printf("On l'a trouvé : \n");
        Afficher(cestLui->pers);
        printf("Confirmez-vous l'ajout de la nouvelle personne ?(O/N) ");
        reponse = toupper(getche());
        }
        if (reponse == 'O' || cestLui == NULL)
             Inserer (P, unePers, avant);
 }
```

```
void Modifier (pointeur * P)
{  pointeur  cestLui, avant, tempo ;
   int aModifier;
   char reponse ;
   personne nouvPers ;
   do
    {
      printf("Quelques numéros existants : 1005, 1101, 1707, 7104, 7506 \n");
      printf("Quelques numéros inexistants : 1899, 2222, 5678\n");
      printf("Entrez le numéro de la personne à modifier ses informations ");
      scanf("%d", &aModifier);
      find (aModifier, *P, &avant, &cestLui );
      reponse = 'N';
      if (cestLui != NULL)
       {
       printf("On l'a trouvé : \n");
       afficher(cestLui->pers);
       printf("\n\n");
       printf("Désirez-vous modifier les informations de cette personne ?(O/N) ");
       fflush(stdin);
       reponse = toupper(getchar());
       if (reponse == 'O')
         { demander( &nouvPers);
           if ( nouvPers.numero == (cestLui->pers).numero )
                 cestLui->pers = nouvPers;
           else
              {
               eliminer ( P, avant, cestLui);
               find ( nouvPers.numero, *P, &avant, &cestLui);
               inserer(P, nouvPers, avant);
              }
         }
       }
      else
       printf("Désolé! on ne trouve pas cette personne\n");
      printf("avez-vous autre personne à modifier (O/N) ? ");
      fflush(stdin);
     }
  while (toupper(getche()) != 'N');
}

void traiter ( char choix, pointeur * L )
/* Dépendant du choix, il est possible que la liste sera modifiée.
   Par cette raison, le passage de la liste est par pointeur.   */
{
  switch(choix)

  { case 'V' : visualiser( *L );
               break;
    case 'C' : chercher ( *L );
               break ;
    case 'S' : supprimer (L);
               break;
    case 'A' : ajouter (L);
               break;
    case 'M' : modifier(L);
               break;
```

```
    case 'I' : inverser(L); /* inverser la liste */
               visualiser(*L);
               inverser(L); /* remettre dans l'ordre FIFO */
               break;
  }
}


void main()
{  char choix ; /* option d'un menu */

   creerFIFO (&liste);
   do
    { obtenir(&choix);           /* choix d'une option du menu */
      traiter(choix, &liste) ;   /* traiter le choix avec la liste */
    }
   while (choix != 'Q');
}
```

Exécution :

```
Tapez :
  - Q pour Quitter
  - V pour Visualiser
  - C pour Chercher
  - A pour Ajouter une nouvelle personne
  - M pour Modifier les informations d'une personne
  - S pour Supprimer une personne de la liste
  - I pour visualiser la liste Inverse


Entrez votre choix parmi Q, V, C, A, M, S ou I >> V


Dans l'ordre FIFO:
  1)  1005   1.47  45.81  F
  2)  1101   1.63  61.23  M
  3)  1103   1.70  64.86  M
  4)  1104   1.75  76.20  M
  5)  1107   1.68  57.61  F
  6)  1405   1.55  55.34  F
  7)  1406   1.52  44.91  F
  8)  1407   1.57  51.71  F
  9)  1503   1.68  64.41  F
 10)  1504   1.57  50.80  F
 11)  1506   1.73  66.22  F
 12)  1707   1.65  58.51  F
 13)  1806   1.78  65.32  M
 14)  1807   1.63  60.33  F
 15)  1901   1.57  56.25  M
 16)  1902   1.73  71.67  M
 17)  1905   1.80  82.10  M
 18)  7102   1.73  63.05  F
 19)  7103   1.78  75.75  M
 20)  7104   1.78  74.84  M
```

```
Appuyez sur Entrée pour continuer ou T pour terminer T

Tapez :
  - Q pour Quitter
  - V pour Visualiser
  - C pour Chercher
  - A pour Ajouter une nouvelle personne
  - M pour Modifier les informations d'une personne
  - S pour Supprimer une personne de la liste
  - I pour visualiser la liste Inverse


Entrez votre choix parmi Q, V, C, A, M, S ou I >> C
Quelques numéros existants   : 1005, 1101, 1707, 7104, 7506
Quelques numéros inexistants : 1899, 2222, 5678
Entrez le numéro de la personne à chercher 1707

YOUPI! on l'a trouvé :


Numéro      :   1707
taille      :   1.65
poids       :   58.5
sexe        : Féminin
valeur avant :  1506


avez-vous une autre personne à chercher (O/N) ? N


Tapez :
  - Q pour Quitter
  - V pour Visualiser
  - C pour Chercher
  - A pour Ajouter une nouvelle personne
  - M pour Modifier les informations d'une personne
  - S pour Supprimer une personne de la liste
  - I pour visualiser la liste Inverse


Entrez votre choix parmi Q, V, C, A, M, S ou I >> S
Quelques numéros existants   : 1005, 1101, 1707, 7104, 7506
Quelques numéros inexistants : 1899, 2222, 5678


Entrez le numéro à supprimer 1101

On l'a trouvé :

Numéro      :   1101
taille      :   1.63
poids       :   61.2
sexe        : Masculin
Confirmez-vous la suppression ?(O/N) O
```

```
avez-vous autre personne à supprimer (O/N) ? o

Quelques numéros existants   : 1005, 1101, 1707, 7104, 7506
Quelques numéros inexistants : 1899, 2222, 5678


Entrez le numéro à supprimer 7104

On l'a trouvé :

Numéro      :   7104
taille      :   1.78
poids       :   74.8
sexe        : Masculin
Confirmez-vous la suppression ?(O/N) o

avez-vous autre personne à supprimer (O/N) ? o

Quelques numéros existants   : 1005, 1101, 1707, 7104, 7506
Quelques numéros inexistants : 1899, 2222, 5678


Entrez le numéro à supprimer 7606

On l'a trouvé :

Numéro      :   7606
taille      :   1.65
poids       :   59.9
sexe        : Féminin
Confirmez-vous la suppression ?(O/N) o

avez-vous autre personne à supprimer (O/N) ? n

Tapez :
  - Q pour Quitter
  - V pour Visualiser
  - C pour Chercher
  - A pour Ajouter une nouvelle personne
  - M pour Modifier les informations d'une personne
  - S pour Supprimer une personne de la liste
  - I pour visualiser la liste Inverse


Entrez votre choix parmi Q, V, C, A, M, S ou I >> M
Quelques numéros existants   : 1005, 1101, 1707, 7104, 7506
Quelques numéros inexistants : 1899, 2222, 5678
Entrez le numéro de la personne à modifier ses informations 1005

On l'a trouvé :


Numéro      :   1005
taille      :   1.47
poids       :   45.8
sexe        : Féminin
```

```
Désirez-vous modifier les informations de cette personne ?(O/N) o

Entrez les informations de la personne :

 - Numéro :   1005
 - sexe   :   f
 - taille :   1.67
 - poids  :   48.5

avez-vous autre personne à modifier (O/N) ? o

Quelques numéros existants   : 1005, 1101, 1707, 7104, 7506
Quelques numéros inexistants : 1899, 2222, 5678

Entrez le numéro de la personne à modifier ses informations 7104


Désolé! on ne trouve pas cette personne

avez-vous autre personne à modifier (O/N) ? n


Tapez :
  - Q pour Quitter
  - V pour Visualiser
  - C pour Chercher
  - A pour Ajouter une nouvelle personne
  - M pour Modifier les informations d'une personne
  - S pour Supprimer une personne de la liste
  - I pour visualiser la liste Inverse


Entrez votre choix parmi Q, V, C, A, M, S ou I >> A

Entrez les informations de la personne :

 - Numéro : 2345
 - sexe   : m
 - taille : 1.67
 - poids  : 67.8

Tapez :
  - Q pour Quitter
  - V pour Visualiser
  - C pour Chercher
  - A pour Ajouter une nouvelle personne
  - M pour Modifier les informations d'une personne
  - S pour Supprimer une personne de la liste
  - I pour visualiser la liste Inverse


Entrez votre choix parmi Q, V, C, A, M, S ou I >> I

Dans l'ordre FIFO:
  1)  7605   1.78  75.75  M
  2)  7604   1.45  40.37  F
```

```
  3)  7603   1.60  48.99  F
  4)  7602   1.65  53.07  F
  5)  7601   1.63  47.63  F
  6)  7507   1.73  72.57  M
  7)  7506   1.68  54.43  F
  8)  7505   1.73  65.32  M
  9)  7502   1.70  60.78  F
 10)  7501   1.65  61.69  M
 11)  7407   1.75  71.67  M
 12)  7405   1.70  63.50  F
 13)  7404   1.83  86.18  M
 14)  7403   1.80  79.38  M
 15)  7402   1.63  53.52  F
 16)  7401   1.63  54.88  F
 17)  7307   1.70  56.25  F
 18)  7305   1.80  89.36  M
 19)  7304   1.65  64.41  M
 20)  7303   1.75  63.96  F


Appuyez sur Entrée pour continuer ou T pour terminer T

Tapez :
  - Q pour Quitter
  - V pour Visualiser
  - C pour Chercher
  - A pour Ajouter une nouvelle personne
  - M pour Modifier les informations d'une personne
  - S pour Supprimer une personne de la liste
  - I pour visualiser la liste Inverse


Entrez votre choix parmi Q, V, C, A, M, S ou I >> Q
```