TDs : Pointeurs

Exercice 1

Soit le morceau de programme suivant :

```
int a;
int *pa;
double x;
int **p;
a=8;
pa=&a;
x=3.14159;
p=&pa;
**p=281;
```

En supposant que la mémoire soit structurée en octets, que les entiers soient codés sur 4 octets, les pointeurs sur 4 octets et que la zone de mémoire automatique soit située en début d'exécution à l'adresse 2004, représentez la mémoire finale de ce programme.

Corrigé

Après l'exécution de ce programme, la mémoire ressemblera au schéma suivant :

variable	adresse mémoire	contenu
a	2004 2005 2006 2007	281
pa	2008 2009 2010 2011	2004
x	2012 2013 2014 2015 2016 2017 2018 2019	3.14159
p	2020 2021 2022 2023	2008

WWW.TALIB24.COM

La dernière instruction (**p=281) sera évaluée comme suit :

- l'affectation =281; va mettre 281 en mémoire. Mais où ?
- p vaut 2008
- *p est la mémoire dont l'adresse est la valeur de p (2008), donc *p est le pointeur d'entier (p est un int **, donc *p est un int *) situé en 2008 et sa valeur est 2004

Exercice 2

A. Soit la fonction suivante :

```
void echange1 (int x, int y)
{ int z;
z=x;
x=y;
y=z;}
```

Pourquoi ne fonctionne-t-elle pas lorsqu'on l'appelle avec par exemple ?
Représentez la mémoire lors de l'exécution de ce morceau de programme.

Corrigé

Les instructions a=2;b=3; mettent 2 et 3 dans deux zones mémoires de type int (supposées allouées préalablement par une déclaration)

int a	int b
2	3

L'appel `echange1(a,b);` alloue automatiquement deux zones mémoires de type `Int` (les paramètres formels `x` et `y` de la fonction `echange1`) et recopie les valeurs de `a` et `b` dans ces zones mémoires :

int a	int b	int x	int y
2	3	2	3

La déclaration `int z;` alloue automatiquement une zone mémoire de type `int` non initialisée (contenu quelconque marqué ? ? ? ?) :

int a	int b	int x	int y	int z
2	3	2	3	??

L'instruction `z=x;` recopie le contenu de `x` dans `z`:

int a	int b	int x	int y	int z
2	3	2	3	2

L'instruction `x=y;` recopie le contenu de `y` dans `x` :

int a	int b	int x	int y	int z
2	3	3	3	2

L'instruction `y=z;` recopie le contenu de `z` dans `y`:

int a	int b	int x	int y	int z
2	3	3	2	2

WWW.TALIB24.COM

Les variables x et y ont bien été permutées, mais notez bien que les originaux a et b sont restés inchangés. Sur l'accolade fermante (fin d'exécution de la fonction echange1, les mémoires allouées automatiquement pour la fonction sont restituées. Résultat des courses : on se retrouve dans la même situation qu'avant l'appel :

int a	int b
2	3

Tout ce travail pour rien...

B. Soit la fonction suivante :

```
void echange2 (int *x, int *y)
{ int *z;
*z=*x;
*x=*y;
*y=*z;}
```

Pourquoi risque-t-elle ne pas fonctionner lorsqu'on l'appelle avec par exemple ?
Représentez la mémoire lors de l'exécution de ce morceau de programme.

Corrigé

Les instructions `a=2;b=3;` mettent 2 et 3 dans deux zones mémoires de type `int` (supposées allouées préalablement par une déclaration et situées par exemple aux adresses 2008 et 2012)

2008 int a	2012 int b
2	3

L'appel `echange2(&a,&b);` alloue automatiquement deux zones mémoires de type `int*` (les paramètres formels x et y de la fonction echange2) et recopie les valeurs de &a (2008) et &b (2012) dans ces zones mémoires :

2008 int a	2012 int b	int *x	int *y
2	3	2008	2012

La déclaration `int *z;` alloue automatiquement une zone mémoire de type `int *` non initialisée (contenu quelconque marqué ? ? ? ?) :

2008 int a	2012 int b	int *x	int *y	int *z
2	3	2008	2012	????

L'instruction `*z=*x;` recopie la valeur de `*x`(c'est-à-dire la valeur de la zone mémoire dont l'adresse est donnée par x , soit la zone mémoire int située en 2008, donc 2) dans la zone mémoire dont l'adresse est donnée par z. Or, z n'étant pas initialisée, la zone mémoire qu'elle indique peut être située n'importe où. Si l'on a de la chance, elle sera

WWW.TALIB24.COM

interdite en écriture et un message de type `Segmentation fault` nous avertira de l'erreur de programmation, avant l'arrêt du programme. Si l'on est moins chanceux, cette zone mémoire ne sera pas interdite en écriture et le programme,
quoique faux, pourra sembler fonctionner comme dans `echange3` (voir ci-dessous).
Dans tous les cas, ce programme est faux et il ne faut jamais utiliser une variable sans l'avoir initialisée

C. Soit la fonction suivante :
void echange3 (int *x, int *y) { int z;
z=*x;*x=*y;*y=z;
}
et l'appel suivant :
Représentez la mémoire lors de l'exécution de ce morceau de programme.

Corrigé

Les instructions `a=2;b=3;` mettent 2 et 3 dans deux zones mémoires de type `int` (supposées allouées préalablement par une déclaration et situées par exemple aux adresses 2008 et 2012)

2008 int a	2012 int b
2	3

L'appel `echange3(&a,&b);` alloue automatiquement deux zones mémoires de type `int *` (les paramètres formels `x` et `y` de la fonction `echange3`) et recopie les valeurs de `&a` (2008) et `&b` (2012) dans ces zones mémoires :

2008 int a	2012 int b	int *x	int *y
2	3	2008	2012

La déclaration `int z;` alloue automatiquement une zone mémoire de type `int` non initialisée (contenu quelconque marqué `????`) :

2008 int a	2012 int b	int *x	int *y	int z
2	3	2008	2012	????

L'instruction `z=*x;` recopie la valeur de `*x` (c'est-à-dire la valeur de la zone mémoire dont l'adresse est donnée par `x`, soit la zone mémoire `int` située en 2008, donc 2) dans la zone mémoire `z` de type `int`:

2008 int a	2012 int b	int x	int y	int z
2	3	2008	2012	2

L'instruction `*x=*y;` recopie le contenu de `*y` (soit le contenu de la zone mémoire de type `int` dont l'adresse est donnée par la valeur de `y`, soit 3), dans`*x` (soit la zone

WWW.TALIB24.COM

mémoire de type `int` dont l'adresse est donnée par la valeur de `x`, soit la zone mémoire située en 2008, c'est à dire `a`) :

2008 int a	2012 int b	int*x	int *y	int z
3	3	2008	2012	2

L'instruction `*y=z;` recopie le contenu de `z` (2) dans `*y` (soit la zone mémoire de type `int` dont l'adresse est donnée par la valeur de `y`, soit la zone mémoire située en 2012, c'est à dire `b`) :

2008 int a	2012 int b	int *x	int *y	int z
3	2	2008	2012	2

Les variables `a` et `b` ont bien été permutées. Sur l'accolade fermante (fin d'exécution de la fonction `echange3`, les mémoires allouées automatiquement pour la fonction sont restituées. Résultat des courses : la fonction `echange3` a bien fonctionné.

2008 int a	2012 int b
3	2

Ouf !

Exercice 3

Soit les deux déclarations suivantes :
char tabString[]="toto";
char *ptrString="Titi";
Essayez d'en trouver les différences de comportement. En particulier, notez les possibilités d'exécuter les instructions suivantes :
tabString[2]='e';
ptrString[2]='e';
tabString=ptrString;
ptrString = tabString;
Lesquelles provoquent une erreur à la compilation ? à l'exécution ? Faites un dessin représentant la mémoire.

Corrigé

tabString[2]='e'; // possible
ptrString[2]='e'; // impossible
tabString=ptrString; // impossible
ptrString = tabString; // possible

Exercice 4

1. Créez une fonction newTriangularMatrix prenant en paramètre une dimension n et renvoyant une matrice triangulaire sous la forme d'un tableau dynamique d'entiers. On ne stockera que les coefficients de la partie supérieure de la matrice :

WWW.TALIB24.COM

$$A = (a_{i,j}) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & \cdots & a_{1,n} \\ 0 & a_{2,2} & & & a_{2,n} \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & a_{n,n} \end{pmatrix}$$

Corrigé

```
int **  newTriangularMatrix(int n)
{       int k=1;
        int ** matrice;
        matrice = (int**)malloc(n * sizeof(int*));
        for (i = 0; i < k; i++)
        {  matrice[i] = (int*)malloc(k * sizeof(int));
           k++;
        }
        return matrice;
}
```

2. Créez une fonction **displayTriangularMatrix** prenant en entrée une matrice triangulaire et l'affichant comme ci-dessus.

```
            1
          1   1
        1   2   1
      1   3   3   1
    1   4   6   4   1
  1   5  10  10   5   1
```

Corrigé

```
void  displayTriangularMatrix (int nbLignes, int Matrice)
{           int  i,  j;         // Indices de boucle
            for  (i=0;  i<=nbLignes;  i++)
            {         // Centrage des lignes
                      for(j=1;  j<=(nbLignes-i)/2;  j++)  printf("    ");
                      for(j=1;  j<=  3*((i+nbLignes)%2);  j++)  printf(" ");
                      // Boucle de la deuxième colonne à la fin de la ligne
                      for(j=0;  j<=i;  j++)
                                printf("%d",  matrice[i][j]);          // Nouvelle ligne
                      printf("\n");
            }
}
```

WWW.TALIB24.COM

3. Créez une fonction **triangleDePascal** prenant en paramètre un nombre de ligne L et renvoyant une matrice triangulaire contenant un triangle de Pascal :

Corrigé

```
Int ** triangleDePascal(int L)
{
int** matrice; /* matrice résultat  */
int i, j;      /* indices courants  */
matrice = newTriangularMatrix(L);
/* */
    for  (i=0;  i<=L;  i++)
    {
          matrice[i][0]=1;    // La première colonne est gérée
          // Boucle de la deuxième colonne à la fin de la ligne
          for(j=1;  j<=i;  j++)
          {       matrice[i][j]= matrice[i][j-1]*(i-j+1)/(j);
          }
    }
}
```

WWW.TALIB24.COM

TD 2 : Chaines de caractères et tableaux

Exercice 1 :

Écrivez une fonction qui prend en argument une chaîne de caractères, la renverse sur elle même ("toto"–→"otot") et retourne l'adresse de cette chaîne.
Prototype : char * miroir (char *s);

Corrigé

Il faut déjà parcourir la chaîne pour accéder à son dernier caractère. Puis permuter les couples de caractères symétriques en évitant de le faire deux fois, ce qui laisserait la chaîne inchangée.

```
char * miroir (char * s)
 { int g,d; /* indice gauche et droit de parcours */
   char c;
   for (d=0;s[d]!=0;++d);

    for (g=0,--d;g<d;++g,--d)
    { c=s[g];
      s[g]=s[d];
      s[d]=c;
     }
    return s;
 }
```

Exercice 2 :

Écrivez une fonction qui prend en argument une chaîne de caractères et l'affiche en répétant chaque caractère n fois (l'appel avec "toto" et 3 affichera "tttoootttooo").
Prototype : void repete (char *s, int n);

Corrigé

```
void repete (char *s, int n)
{         int j;
        for(;*s!='\0';++s)
        { for (j=0;j<n;++j)
                {
                 printf("%c",*s);
                }
        }
}
```

Exercice 3 :

Écrivez une fonction qui prend en argument une chaîne de caractères, la transforme en majuscules ("toto"–→"TOTO") et retourne son adresse. On laissera inchangés les caractères non lettre. On supposera la chaîne sans caractères accentués ou cédillés.
Prototype : char * majuscule (char *s);

Corrigé

L'écart entre majuscules et minuscules en ASCII est de 32 ((20)hexa). Ce n'est pas la peine de

WWW.TALIB24.COM

s'en souvenir : 'a'-'A'—→32

```
char * majuscule (char *s)
{        char *ret = s;
       char * majuscule (char *s);
       for(;*s!='\0';++s)
       { if ('a'<=*s && *s<='z')
         { *s += 'A'-'a'
         }
       }
return ret;
}
```

Exercice 4 :

Écrivez une fonction qui prend en argument deux chaînes de caractères et retourne 1 si la première chaîne commence par la seconde (les premiers caractères de la chaîne1 sont ceux de la chaîne2) et 0 sinon. Écrivez un programme pour tester cette fonction.
Prototype : int debute_par (char * chaine1, char * chaine2);

Corrigé

Il suffit de regarder pour chaque caractère de chaine2 s'il est égal au caractère correspondant de chaine1. Dès qu'il y a inégalité, on retourne 0 (faux). Si on arrive en bout de chaine2 sans avoir détecté d'inégalité, on retourne 1 (vrai).

Note : ce programme renvoie toujours 1 si la chaine2 est vide.

```
int debute_par (char * chaine1, char * chaine2)
{       int i;
       for (i=0;chaine2[i]!='\0';++i)
              if (chaine1[i]!=chaine2[i])
                 return 0;
         return 1;
}
```

Exercice 5 :

Écrivez une fonction qui prend en argument deux chaînes de caractères et retourne la position de la première occurrence de la chaîne2 dans la chaîne1 si elle y est présente et -1 sinon.
Prototype : int presence (char * chaine1, char * chaine2);

Corrigé

Un appel itératif à debute_par permettra de détecter la présence de la sous-chaîne.
Note : ce programme considère que la chaîne vide débute toute chaîne (retour 0).

```
int presence (char * chaine1, char * chaine2)
{      int i;
       for (i=0;chaine1[i]!='\0';++i)
              if (debute_par(&chaine1[i],chaine2))
                 return i;
         return -1;
}
```

WWW.TALIB24.COM

Exercice 6 :

Écrivez une fonction qui compte le nombre d'occurrences d'un caractère c dans une chaîne s. La fonction devra être récursive. Écrivez un programme pour tester cette fonction.

Prototype : int compte (char c, char * s)

Corrigé

```
int compte (char c, char * s)
{   if (*s=='\0')
        return 0;
return compte(c,s+1) + (*s==c?1:0) ;
}
```

Exercice 7 :

Écrivez une fonction qui recherche dans une chaîne chaque caractère c pour le remplacer par un caractère r et retourne l'adresse de la chaîne.

Prototype : char * cherche_remplace (char c, char r, char * s);

Corrigé

C'est un parcours simple avec remplacement lorsque l'on tombe sur le caractère à remplacer :

```
char * cherche_remplace (char c, char r, char * s)
{       int i;
        for (i=0;s[i]!='\0';++i)
                if (s[i]==c)
                        s[i]=r;
        return(s);
}
```

WWW.TALIB24.COM

TD & TP 3 : Fonctions
Programmation en Langage C

Consignes :

- **Essayer d'utiliser la récursivité à la place des boucles.**

Exercice 1 :

La suite des nombres de Fibonacci est définie par ses deux premiers termes F0 =0,F1 =1
Pourtout n≥0 Fn+2 =Fn+1 +Fn

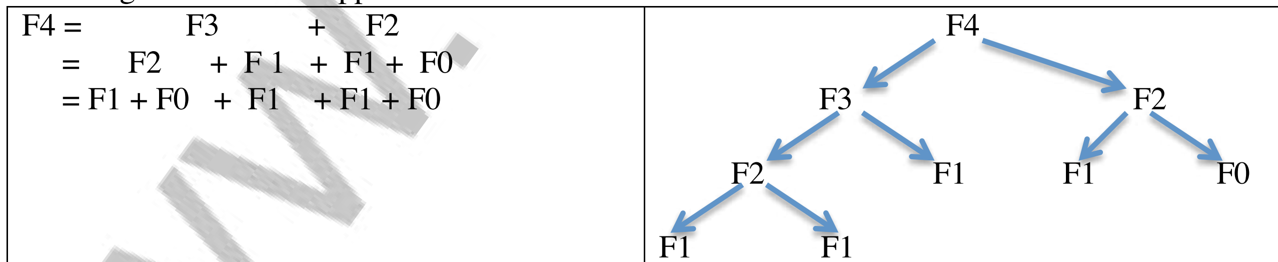
1. Réalisez une fonction nommée ***fibonacci*** paramétrée par un entier n qui calcule le nombre Fn.
2. Dessinez l'arbre des appels à la fonction ***fibonacci*** dans le calcul de ***fibonacci(4).***
3. Trouver une relation de récurrence pour le nombre d'appels à la fonction ***fibonacci*** pour calculer Fn. Puis programmez le calcul de ce nombre.

<u>Corrigé</u>

1- Il s'agit de donner une solution sans récursivité

```
int fibonacci (int n)
{
  int nbren=1,i,som=0, tmp;
  if (n==0 ) return 0
  else if (n==1) returm 1
  else
   { for(i=1;i<=n;i++)
         {
          tmp = som;
         som = som + nbren;
         nbren = nbren + tmp;
          }
     return(nbren);
   }
}
```

2- il s'agit de tracer les appels selon la formule Fn+2 =Fn+1 +Fn



3- ***fibonacci(n)= fibonacci(n-2) +- fibonacci(n)***

```
int fibonacci (int n)
{
  int nbren=1,i,som=0, tmp;
  if (n==0 ) return 0
```

WWW.TALIB24.COM

```
  else if (n==1) returm 1
  else
    return(fibonacci(n-2)+ fibonacci(n-1));
}
```

Exercice 2 :

Ecrire une fonction qui reçoit comme paramètre une liste d'entiers dans un tableau d'entiers et parcourt cette liste en comptant les répétitions successives des valeurs. La fonction retournera un tableau à deux colonnes dont la première contient les valeurs de la liste dans leur ordre d'apparition et la seconde leur nombre de répétitions successives.

Corrigé

```
int ** Occurrences(int * Tab, int n)
{      int i,j, nombreOccu=1;
        int ** OccuTab ;
        OccuTab = (int**)malloc(n * sizeof(int*));
        for (i = 0; i < n; i++)
             {  OccuTab [i] = (int*)malloc(2 * sizeof(int));
             }
        for (i = 0; i < n; i++)
        {  nombreOccu=1 ;
               for (j=i+1 ; j<n ; j++)
               {   if (Tab[i]==Tab[j])
                          nombreOccu++ ;
                }
             OccuTab[i][0]= Tab[i];
             OccuTab[i][0]= nombreOccu ;
             Tab[i]=-1;
        }
        return OccuTab ;
}
```

Exercice 3 :

Ecrire, en utilisant deux méthodes différentes, une fonction qui calcule la somme des chiffres d'un nombre entier strictement positif et recommence le calcul avec le résultat obtenu tant que celui-ci n'est pas compris entre 1 et 9. Après chaque calcul la fonction affiche à l'écran la somme obtenue. La fonction retournera le nombre entre 1 et 9 obtenu

Corrigé 1

```
Void somme (int N)
{       int n = N;
        while (n >9 )
           { printf( ‘’ %d  = %d +%d‘’, n , n-(n%10), n%10) ;
                n= n-(n%10) ;
           }
}
```

WWW.TALIB24.COM

Corrigé 2

```
Void somme (int N)
{
        if (N>0 || N<10)
            {printf( '' %d  = %d +%d'',N , N-(N%10), N%10) ;
             somme(N-(N%10) );
            }
}
```

Exercice 4 :

Le but de cet exercice sera de vous familiariser à la gestion de *argc* et *argv*. Le principe est simple : afficher dans notre programme les arguments qui lui ont été passés par la console.

Corrigé

```
int main (int argc, char * argv[])
{
int i;
if (argc < 2)
{
printf("\nErreur : nombre invalide d'arguments");
return(EXIT_FAILURE);     /*ou bien exit(EXIT_FAILURE);*/
}
for (i=1; i<argc; i++)
{  printf("\n l'argument numéro  %d   est %c: %d\n",i,  *argv[i]);
}
return(EXIT_SUCCESS);       /*ou bien exit(EXIT_SUCCESS);*/
}
```

Exercice 5 :

Écrire un programme addition, dont le but sera d'additionner les deux arguments passés au programme à la fonction main.

Corrigé

```
int main (int argc, char *argv[])
{
int a, b;
if (argc != 3)
{
printf("\nErreur : nombre invalide d'arguments");
return(EXIT_FAILURE);     /*ou bien exit(EXIT_FAILURE);*/
}
```

WWW.TALIB24.COM

```
a = atoi(argv[1]);
b = atoi(argv[2]);
printf("\nLa somme de %d par %d vaut : %d\n", a, b, a +b);
return(EXIT_SUCCESS);       /*ou bien exit(EXIT_SUCCESS);*/
}
```

WWW.TALIB24.COM

TD & TP 4
Pointeurs de fonctions et nombre de paramètres variables
Programmation en Langage C

Exercice 1 :

1. Écrire une fonction multDeux qui multiplie par deux un entier.
2. Écrire une fonction qui étant donné une fonction f prenant un entier en argument et un tableau t d'entiers, applique la fonction f aux éléments de t.
3. Écrire la fonction main qui applique multDeux à un tableau d'entiers.

Corrigé :

1.
```
Int  multDeux(int  n)
{
return n*2;
}
```

2.
```
void Applique(int (*f)(int), int *Tab)
{
int i;
for (i=0;i<(sizeof(Tab)/sizeof(int)); i++)
        printf("\n L'application de la fonction aux %d ème élément du tableau est %d", i,
f(Tab[i]));
}
```

3.
```
void main()
{int *T;
int n=0;
printf("\n entrer le nombre d'élément  du tableau ");
scanf("%d",&n);
T = (int*)malloc(n * sizeof(int));
     for (i = 0; i < n; i++)
              {  printf("\n entrer le %d ème élément  du tableau ");
                 scanf("%d",&T[i]);
          }

Applique(multDeux, T);
}
```

WWW.TALIB24.COM

Exercice 2 :

Écrire une fonction qui calcule la moyenne d'un nombre variable d'entiers passés en paramètre.

Corrigé :

```
int Moyenne(int n, ...)
{
int res = 0,i;
va_list liste_parametres;
va_start(liste_parametres, n);
for (i = 0; i < n; i++)
        res += va_arg(liste_parametres, int);
va_end(liste_parametres);
return(res/n);


}
```

WWW.TALIB24.COM

TP5: Structures

Exercice 1 :

Dans cet exercice, nous allons manipuler une structure représentant un étudiant qui comprendra son nom, son prénom et une note. 1. Pour la première version de cet exercice, nous supposerons que nous manipulerons un nombre fini d'étudiants représenté par la constante NMAX.
La structure utilisée sera la suivante :

```
typedef struct etudiant { char * nom;
char * prenom;
int note; } etudiant;
```

Comme on peut le voir sur cette structure, les champs `nom` et `prenom` sont des pointeurs sur des chaînes de caractères non allouées car on souhaite utiliser juste la mémoire nécessaire pour le nom et le prénom. Il sera donc nécessaire de réaliser une allocation dynamique de mémoire pour les champs `nom` et `prenom`.
On travaillera sur un groupe d'étudiants dont la définition est `etudiant groupe[NMAX] ;`

1. Écrire la fonction saisir dont l'en-tête est `void saisir(etudiant *e)` permettant la saisie d'un étudiant ;

2. Écrire la fonction afficher permettant l'affichage d'un étudiant passé en paramètre ; c. écrire la fonction moyenne dont l'en-tête est `int moyenne(etudiant e[])` retournant la moyenne de tous les étudiants.

WWW.TALIB24.COM

Corrigé

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

#define NMAX 2

typedef struct  {
    char * nom;
    char * prenom;
    int note;
}  etudiant;




// saisie d'un Ã©tudiant
void saisir(etudiant *e)
{
    char msg[256];
    int n;

    printf("\nEntrez le nom de l'Ã©tudiant ");
    scanf("%s",msg);
    n = strlen(msg);

    e->nom = (char *) malloc ((n+1)*sizeof(char));
    strcpy(e->nom,msg);

    printf("\nEntrez le prÃ©nom de l'Ã©tudiant ");
    scanf("%s",msg);

    e->prenom = (char *) malloc ((n+1)*sizeof(char));
```

WWW.TALIB24.COM

```
    strcpy(e->prenom,msg);

    printf("\n Entrez la note de l'Ã©tudiant ");
    scanf("%d",&(e->note));
}

//affichage de la moyenne
void affichage(etudiant e)
{
    printf("\n%s\t%s\t%d\n",e.nom,e.prenom,e.note);
}

// calcul de la moyenne
int moyenne(etudiant e[])
{
    int somme = 0;
    int i;

    for(i = 0; i < NMAX; i++)
        somme += e[i].note;

    return somme / NMAX;
}

main()
{
    etudiant groupe[NMAX];
    etudiant * et;

    et = groupe;

    int i;
    for(i = 0; i < NMAX; i++){
        saisir(et);
        et++;
    }

    et = groupe;
    for(i = 0; i < NMAX; i++){
        affichage(* et);
        et++;
    }

    printf("\n moyenne -> %d",moyenne(groupe));
}
```

WWW.TALIB24.COM

Dans la réalité, le nombre d'étudiants fluctue tout au long de l'année. Nous allons adapter la question précédente avec cette nouvelle contrainte en utilisant pour représenter les étudiants une liste chaînée. La définition de notre structure `etudiant` sera donc :

```
typedef struct etudiant { char * nom;
char * prenom; int note;
struct etudiant *suivant;
} etudiant;
```

1. Écrire une fonction `insere_en_tete` dont l'en-tête est `void insere_en_tete(etudiant ** classe)` permettant d'insérer en tête de la liste chaînée un étudiant. Il faudra également réaliser une allocation dynamique pour la saisie de l'étudiant

2. Écrire la fonction `void affichage(etudiant * groupe)` affichant tous les étudiants saisis jusqu'à maintenant ;

3. Écrire la fonction moyenne dont l'en-tête est `int moyenne(etudiant * groupe)` retournant la moyenne de tous les étudiants.

WWW.TALIB24.COM

Corrigé

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

#define NMAX 2

typedef struct  etudiant {
    char * nom;
    char * prenom;
    int note;
    struct etudiant *suivant;
}  etudiant;




// saisie d'un Ã©tudiant
void insere_en_tete(etudiant ** classe)
{
    char msg[256];
    int n;
    etudiant e, *pe;

    pe = (etudiant *) malloc (sizeof(etudiant));


    printf("\nEntrez␣le␣nom␣de␣l'Ã©tudiant␣");
    scanf("%s",msg);
    n = strlen(msg);

    pe->nom = (char *) malloc ((n+1)*sizeof(char));
    strcpy(pe->nom,msg);

    printf("\nEntrez␣le␣prÃ©nom␣de␣l'Ã©tudiant␣");
    scanf("%s",msg);

    pe->prenom = (char *) malloc ((n+1)*sizeof(char))
    strcpy(pe->prenom,msg);

    printf("\n␣Entrez␣la␣note␣de␣l'Ã©tudiant␣");
    scanf("%d",&(pe->note));

    pe->suivant = *classe;
    *classe = pe;
```

WWW.TALIB24.COM

```
}

//affichage de la moyenne
void affichage(etudiant * groupe)
{
    etudiant e;

    while (groupe != NULL){
        e = * groupe;
        printf("\n%s\t%s\t%d\n",e.nom,e.prenom,e.note);
        groupe = groupe->suivant;
    }
}

// calcul de la moyenne
int moyenne(etudiant * groupe)
{
    int somme = 0;
    int i;

    for(i = 0; i < NMAX; i++)
        somme += e[i].note;

    return somme / NMAX;
}

main()
{
    etudiant * groupe;

    groupe = NULL;

    insere_en_tete(&groupe);
    insere_en_tete(&groupe);
    insere_en_tete(&groupe);
    affichage(groupe);

}
```

WWW.TALIB24.COM

Exercice 2 :

1. Ecrire une structure décrivant une carte grise avec les éléments suivants : nom et prénom du propriétaire, numéro d'immatriculation, puissance fiscale, date de mise en service. Afficher à l'écran l'ensemble des données de la carte grise que vous avez saisie au clavier (le champ date pourra être considéré comme un tableau de 3 entiers).

2. Reprendre la question précédente en utilisant maintenant une structure `Date` pour modéliser la date de mise en circulation. On aura donc 3 champs entiers (`int`) dans cette structure afin de définir le jour, le mois et l'année. Faire de même l'affichage de toutes les données de la carte grise.

3. Écrire une fonction `comparaison()` qui prend en paramètre deux dates, à l'aide de la structure `Date`, et qui retourne la valeur entière 1 si les dates sont identiques et qui retourne 0 dans les autres cas.

Exercice 3 :

Le but de l'exercice est la réalisation d'un ensemble dont les éléments peuvent être ajoutés et retirés en temps constant. L'ensemble est représenté par une structure incluant un tableau dont les cellules contiennent les adresses des éléments, ainsi qu'un entier ayant pour valeur le nombre d'éléments stockés. Chaque élément est une structure constituée d'une chaîne de caractères et d'un entier appelé indexe qui contient l'indice auquel l'adresse de l'élément est placée dans le tableau. C'est cet indexe qui permet de retirer n'importe quel élément de l'ensemble en temps constant, indépendamment du nombre d'éléments déjà stockés. Vous devez réaliser les fonctions suivantes :

1. Définir les deux structures `element` et `ensemble`

2. Écrire la fonction `element* creeElem(char* s)` qui crée un nouvel élément contenant une *copie* de la chaîne s et retourne l'adresse de l'élément créé. Cet élément doit être stocké dans un bloc mémoire de la taille appropriée réservé dans le tas par la fonction malloc.

3. Écrire la fonction `void ajouteElem(ensemble* w, element* e)` qui ajoute l'élément pointé par e dans l'ensemble pointé par w. On suppose que l'élément à ajouté a été préalablement créé avec la fonction creeElem.

4. Écrire la fonction `void retireElem(ensemble* w, element* e)` qui retire de l'ensemble pointé par w l'élément d'adresse e. On suppose que cet élément est initialement présent dans l'ensemble.

5. Réaliser une fonction `main()` qui crée des éléments contenant les chaînes «aaa», «bbb», «ccc», qui les ajoute à un ensemble initialement vide, puis qui retire l'éléments contenant la chaîne «bbb».

WWW.TALB24.COM

TP-TD 6 Fichiers

Gestion de comptes bancaires

1. définir un type structure **DATE** qui contient trois membres entiers : jour, mois et année
2. définir un type structure **CLIENT** qui contient les membres suivants :
   - numero_cmpt : entier (numéro de compte)
   - nom : chaîne de caractères (nom d'un client)
   - der_operation : caractère (R : Retrait, V : Virement)
   - anc_solde : réel (ancien solde)
   - nouv_solde : réel (nouveau solde)
   - date : DATE (jj mm aa)
3. écrire une fonction **ouvrir** qui ouvre un fichier existant ou le crée sinon
4. écrire une fonction **fermer** qui ferme le fichier
5. écrire la fonction **main** qui fait appel SEULEMENT aux fonctions ouvrir, fermer et la fonction **menu**. Le début de la fonction menu est donné. Avant de continuer VERIFIEZ que le fichier est bien créé
6. écrire une fonction **ajout** qui ajoute un client
7. écrire une fonction **affiche** qui affiche le compte d'un client. Cette fonction doit être, capable de chercher un client soit par son nom soit par son numéro de compte.
   TESTEZ votre programme avant de continuer
8. écrire une fonction **lister** qui affiche tous les comptes des clients
   TESTEZ votre programme avant de continuer
9. écrire une fonction **operation** qui réalise les retraits, les virements et les mises à jour des comptes.

WWW.TALB24.COM

Corrigé

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

#define MAXNOM 20

typedef struct date {
  int jour;
  int mois;
  int annee;
} DATE;

typedef struct client {
  int numero_cpt;
  char nom[MAXNOM];
  char der_operation;
  double anc_solde;
  double nouv_solde;
  DATE date;
} CLIENT;

void getDate(DATE *d) {
  time_t nsec;
  struct tm *temps;
  nsec = time(NULL);
  temps = localtime(&nsec);
  d->jour = temps->tm_mday;
  d->mois = temps->tm_mon + 1;
  d->annee = temps->tm_year + 1900;
}

void ouvrir(FILE **f, char nomfich[]) {
  *f = fopen(nomfich, "r+");
  if (*f == NULL) {
    *f = fopen(nomfich, "w+");
    if (*f == NULL) {
      perror("Erreur d'ouverture du fichier");
      exit(EXIT_FAILURE);
    }
  }
}

void fermer(FILE *fich) {
  if (fich != NULL) {
    fclose(fich);
  }
}

int chercher_nom(FILE *fich, char *nom) {
  CLIENT client;
  int trouve = 0, ret;
```

WWW.TALIB24.COM

```
  rewind(fich);
  while (! trouve) {
    ret = fread(&client, sizeof(CLIENT), 1, fich);
    if (ret == 0) break;
    if (strcmp(client.nom, nom) == 0) {
      fseek(fich, -1l * sizeof(CLIENT), SEEK_CUR);
      return 1;
    }
  }
  return 0;
}

int chercher_compte(FILE *fich, int cpt) {
  CLIENT client;
  int trouve = 0, ret;
  rewind(fich);
  while (! trouve) {
    ret = fread(&client, sizeof(CLIENT), 1, fich);
    if (ret == 0) break;
    if (client.numero_cpt == cpt) {
        fseek(fich, -1l * sizeof(CLIENT), SEEK_CUR);
        return 1;
    }
  }
  return 0;
}

int ajout(FILE *fich) {
  int ret;
  char *pc;
  CLIENT client;
  printf("Ajout d'un client\n");
  printf("\tNumero de compte : ");
  scanf("%d", &client.numero_cpt);
  if (chercher_compte(fich, client.numero_cpt)) {
    fprintf(stderr, "Compte existant\n");
    return 0;
  }
  getchar();
  printf("\tNom : ");
  fgets(client.nom, MAXNOM, stdin); pc = strchr(client.nom, '\n'); *pc = 0;
  if (chercher_nom(fich, client.nom)) {
    fprintf(stderr, "Nom existant\n");
    return 0;
  }
  client.der_operation = 'V';
  client.anc_solde = 0.0;
  printf("\tSolde initial : ");
  scanf("%lf", &client.nouv_solde);
  getDate(&client.date);
  fseek(fich, 0, SEEK_END);
  ret = fwrite(&client, sizeof(CLIENT), 1, fich);
  return ret;
```

WWW.TALIB24.COM

```
}

void affiche(FILE *fich) {
 CLIENT cli;
 char nom[MAXNOM];
 int cpt, ret;
 char choix;
 printf("Consultation par nom ou par compte\n");
 do {
   printf("par nom (n) ou par compte (c) ? ");
   scanf(" %c", &choix);
 } while (choix != 'n' && choix != 'c');
 if (choix == 'c') {
   printf("Numero du compte : ");
   scanf("%d", &cpt);
   ret = chercher_compte(fich, cpt);
 }
 else {
   printf("Nom du compte : ");
   scanf("%s", nom);
   ret = chercher_nom(fich, nom);
 }
 if (ret == 0) {
   printf("Compte ou nom inexistant...\n");
 }
 else {
   fread(&cli, sizeof(CLIENT), 1, fich);
   printf("Compte %d\nNom %s\nDerniere operation %c\nAncien solde %.2f\nNouveau solde %.2f\nDate 
%d/%d/%d\n", cli.numero_cpt,cli.nom,
          cli.der_operation, cli.anc_solde, cli.nouv_solde, cli.date.jour,
          cli.date.mois, cli.date.annee);
 }
}

void lister(FILE *fich) {
 CLIENT cli;
 rewind(fich);
 printf("Listage du contenu du fichier.\n");
 printf("Num\tNom\tOpe\tAnc\tNouv\tDate\n");
 while (fread(&cli, sizeof(CLIENT), 1, fich) == 1) {
   printf("%d\t%s\t%c\t%.2f\t%.2f\t%d/%d/%d\n", cli.numero_cpt,cli.nom,
          cli.der_operation, cli.anc_solde, cli.nouv_solde, cli.date.jour,
          cli.date.mois, cli.date.annee);
 }
}

void operation(FILE *fich) {
 CLIENT cli;
 char choix;
 double somme;
 printf("Numero du compte : ");
 scanf("%d", &cli.numero_cpt);
 if (!chercher_compte(fich, cli.numero_cpt)) {
```

WWW.TALIB24.COM

```
    printf("Compte inexistant...\n");
    return;
  }
  fread(&cli, sizeof(CLIENT), 1, fich);
  printf("Compte %d\nNom %s\nAncien solde %.2f\nNouveau solde %.2f\nDate %d/%d/%d\n",
            cli.numero_cpt,cli.nom, cli.anc_solde, cli.nouv_solde,
            cli.date.jour, cli.date.mois, cli.date.annee);
  printf("Que voulez-vous faire ?\n");
  do {
    printf(" Versement : V\n Retrait   : R\n   Votre choix : ");
    scanf(" %c", &choix);
  } while (choix != 'V' && choix != 'R');
  cli.anc_solde = cli.nouv_solde;
  printf("   Somme : ");
  scanf("%lf", &somme);
  if (somme < 0.0) {
    somme = -somme;
  }
  if (choix == 'V') {
    cli.der_operation = 'V';
    cli.nouv_solde += somme;
  }
  else {
    cli.der_operation = 'R';
    cli.nouv_solde -= somme;
  }
  getDate(&cli.date);
  fseek(fich, -1l * sizeof(CLIENT), SEEK_CUR);
  fwrite(&cli, sizeof(CLIENT), 1, fich);
}

void menu(FILE *fic) {
  char choix;
  do {
    printf("\n\nAjouter d'un nouveau client ...........: A\n");
    printf("Consultation d'un compte client .......: C\n");
    printf("Lister tous les comptes des clients ...: L\n");
    printf("Op√©ration sur un compte client ........: O\n");
    printf("Quitter ..............................: Q\n");
    printf("                          votre choix: ");
    scanf(" %c", &choix);
    printf("\n");
    switch(choix) {
    case 'a':
    case 'A':
      ajout(fic);
      break;
    case 'c':
    case 'C':
      affiche(fic);
      break;
    case 'l':
    case 'L':
```

WWW.TALIB24.COM

```
      lister(fic);
    break;
    case 'o':
    case 'O':
      operation(fic);
    break;
    case 'q':
    case 'Q':
      break;
    default:
      printf("Erreur de saisie, recommencez...\n");
      break;
    }
  }
  while (choix != 'q' && choix != 'Q');
}

int main() {
  FILE *fich;
  ouvrir(&fich, "repert.dat");
  menu(fich);
  fermer(fich);
  return 0;
}
```

WWW.TALIB24.COM