a) Tableaux à une dimension - Vecteurs

Exercice 7.8 Produit scalaire de deux vecteurs

Ecrire un programme qui calcule le produit scalaire de deux vecteurs d'entiers U et V (de même dimension).

Exemple:

```
/           \   /           \
| 3   2   -4 | * | 2   -3   5 | = 3*2+2*(-3)+(-4)*5 = -20
\           /   \           /
```

Exercice 7.9 Calcul d'un polynôme de degré N

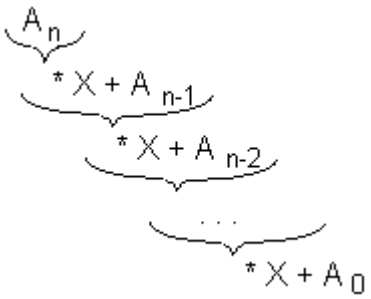
Calculer pour une valeur X donnée du type **float** la valeur numérique d'un polynôme de degré n:

$$P(X) = A_nX^n + A_{n-1}X^{n-1} + ... + A_1X + A_0$$

Les valeurs des coefficients A_n, ... , A_0 seront entrées au clavier et mémorisées dans un tableau A de type **float** et de dimension n+1.

a) Utilisez la fonction **pow()** pour le calcul.

b) Utilisez le *schéma de Horner* qui évite les opérations d'exponentiation:



Exercice 7.10 Maximum et minimum des valeurs d'un tableau

Ecrire un programme qui détermine la plus grande et la plus petite valeur dans un tableau d'entiers A. Afficher ensuite la valeur et la position du maximum et du minimum. Si le tableau contient plusieurs maxima ou minima, le programme retiendra la position du premier maximum ou minimum rencontré.

Exercice 7.11 Insérer une valeur dans un tableau trié

Un tableau A de dimension N+1 contient N valeurs entières triées par ordre croissant; la $(N+1)^{ième}$ valeur est indéfinie. Insérer une valeur VAL donnée au clavier dans le tableau A de manière à obtenir un tableau de N+1 valeurs triées.

Exercice 7.12 Recherche d'une valeur dans un tableau

Problème: Rechercher dans un tableau d'entiers A une valeur VAL entrée au clavier. Afficher la position de VAL si elle se trouve dans le tableau, sinon afficher un message correspondant. La valeur POS qui est utilisée pour mémoriser la position de la valeur dans le tableau, aura la valeur -1 aussi longtemps que VAL n'a pas été trouvée.

Implémenter deux versions:

a) La recherche séquentielle

Comparer successivement les valeurs du tableau avec la valeur donnée.

b) La recherche dichotomique ('recherche binaire', 'binary search')

Condition: *Le tableau A doit être trié*

Comparer le nombre recherché à la valeur au milieu du tableau,

- s'il y a égalité ou si le tableau est épuisé, arrêter le traitement avec un message correspondant.

- si la valeur recherchée précède la valeur actuelle du tableau, continuer la recherche dans le demi-tableau à gauche de la position actuelle.

- si la valeur recherchée suit la valeur actuelle du tableau, continuer la recherche dans le demi-tableau à droite de la position actuelle.

Ecrire le programme pour le cas où le tableau A est trié par ordre croissant.

Question: Quel est l'avantage de la recherche dichotomique? Expliquer brièvement.

Exercice 7.13 Fusion de deux tableaux triés

Problème: On dispose de deux tableaux A et B (de dimensions respectives N et M), triés par ordre croissant. Fusionner les éléments de A et B dans un troisième tableau FUS trié par ordre croissant.

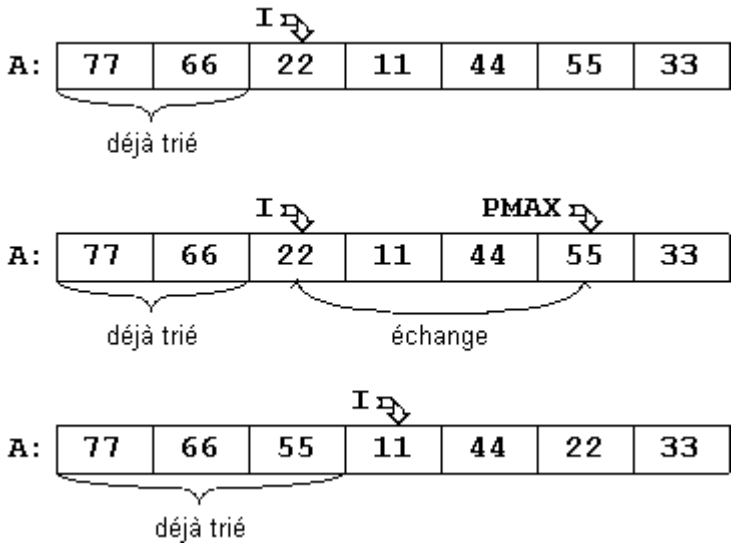
Méthode: Utiliser trois indices IA, IB et IFUS. Comparer A[IA] et B[IB]; remplacer FUS[IFUS] par le plus petit des deux éléments; avancer dans le tableau FUS et dans le tableau qui a contribué son élément. Lorsque l'un des deux tableaux A ou B est épuisé, il suffit de recopier les éléments restants de l'autre tableau dans le tableau FUS.

Exercice 7.14 Tri par sélection du maximum

Problème: Classer les éléments d'un tableau A par ordre décroissant.

Méthode: Parcourir le tableau de gauche à droite à l'aide de l'indice I. Pour chaque élément A[I] du tableau, déterminer la position PMAX du (premier) maximum à droite de A[I] et échanger A[I] et A[PMAX].

Exemple:

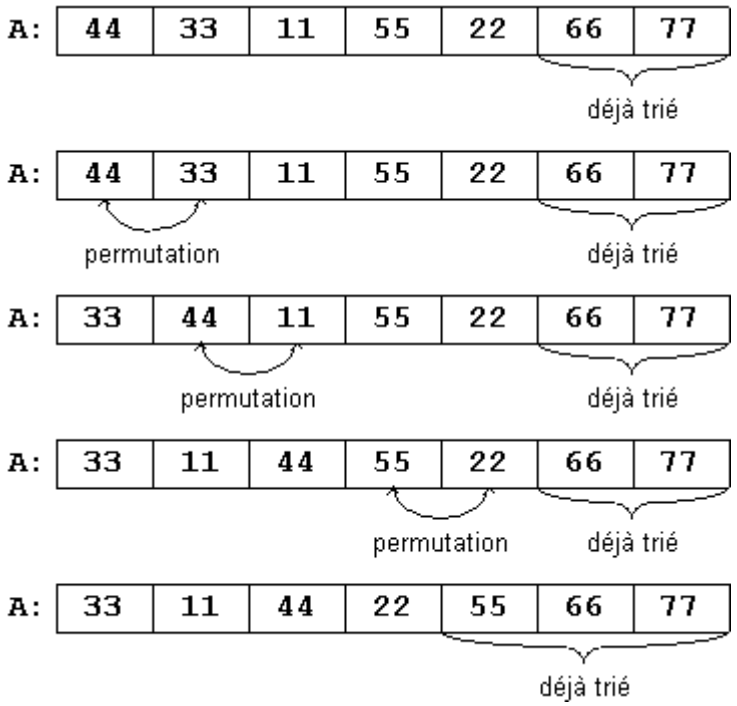


Exercice 7.15 Tri par propagation (bubble sort)

Problème: Classer les éléments d'un tableau A par ordre croissant.

Méthode: En recommençant chaque fois au début du tableau, on effectue à plusieurs reprises le traitement suivant: On propage, par permutations successives, le plus grand élément du tableau vers la fin du tableau (comme une bulle qui remonte à la surface d'un liquide).

Exemple:



Implémenter l'algorithme en considérant que:

* La partie du tableau (à droite) où il n'y a pas eu de permutations est triée.

* Si aucune permutation n'a eu lieu, le tableau est trié.

Exercice 7.16 Statistique des notes

Ecrire un programme qui lit les points de N élèves d'une classe dans un devoir et les mémorise dans un tableau POINTS de dimension N.

* Rechercher et afficher:

- la note maximale,

- la note minimale,

- la moyenne des notes.

* A partir des POINTS des élèves, établir un tableau NOTES de dimension 7 qui est composé de la façon suivante:

`NOTES[6]` contient le nombre de notes 60

`NOTES[5]` contient le nombre de notes de 50 à 59

`NOTES[4]` contient le nombre de notes de 40 à 49

`...`

`NOTES[0]` contient le nombre de notes de 0 à 9

Etablir un graphique de barreaux représentant le tableau NOTES. Utilisez les symboles ####### pour la représentation des barreaux et affichez le domaine des notes en dessous du graphique.

Idée: Déterminer la valeur maximale MAXN dans le tableau NOTES et afficher autant de lignes sur l'écran. (Dans l'exemple ci-dessous, MAXN = 6).

Exemple:

```
La note maximale est 58
La note minimale est 13
La moyenne des notes est 37.250000


  6  >                              #######
  5  >                        ####### #######
  4  >                  ####### ####### #######
  3  >                  ####### ####### ####### #######
  2  >          ####### ####### ####### ####### #######
  1  >          ####### ####### ####### ####### #######
      +-------+-------+-------+-------+-------+-------+-------+
      I 0 - 9 I 10-19 I 20-29 I 30-39 I 40-49 I 50-59 I  60   I
```

Correction tableau:

Exercice 7.8 Produit scalaire de deux vecteurs

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int U[50], V[50]; /* tableaux donnés */
 int N;     /* dimension        */
 int I;     /* indice courant   */
 long PS;   /* produit scalaire */
 /* Saisie des données */
 printf("Dimension des tableaux (max.50) : ");
 scanf("%d", &N );
 printf("** Premier tableau **\n");
 for (I=0; I<N; I++)
    {
     printf("Elément %d : ", I);
     scanf("%d", &U[I]);
    }
 printf("** Deuxième tableau **\n");
 for (I=0; I<N; I++)
    {
     printf("Elément %d : ", I);
     scanf("%d", &V[I]);
    }
 /* Calcul du produit scalaire */
```

```
 for (PS=0, I=0; I<N; I++)
      PS += (long)U[I]*V[I];
  /* Edition du résultat */
 printf("Produit scalaire : %ld\n", PS);
 return 0;
}
```

Exercice 7.9 Calcul d'un polynôme de degré N

Solution combinée:

```
#include <stdio.h>
#include <math.h>
main()
{
 float A[20];/* tableau des coefficients de P */
 int I;      /* indice courant    */
 int N;      /* degré du polynôme */
 float X;    /* argument */
 float P;    /* résultat */

 /* Saisie du degré N et de l'argument X */
 printf("Entrer le degré  N du polynôme (max.20) : ");
 scanf("%d", &N);
 printf("Entrer la valeur X de l'argument : ");
 scanf("%f", &X);
 /* Saisie des coefficients */
 for (I=0 ; I<=N ; I++)
    {
     printf("Entrer le coefficient A%d : ", I);
     scanf("%f", &A[I]);
    }

 /* a) Calcul à l'aide de pow
 for (P=0.0, I=0 ; I<=N ; I++)
       P +=  A[I]*pow(X,I);      */

 /* b) Calcul de Horner */
 /* commencer le calcul avec le dernier coefficient...*/
 for (P=0.0, I=0 ; I<=N ; I++)
       P = P*X + A[N-I];

 /* Edition du résultat */
 printf("Valeur du polynôme pour X = %.2f : %.2f\n", X, P);
 return 0;
}
```

Exercice 7.10 Maximum et minimum des valeurs d'un tableau

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50]; /* tableau donné  */
```

```
 int N;     /* dimension      */
 int I;     /* indice courant */
 int MIN;   /* position du minimum */
 int MAX;   /* position du maximum */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
    {
     printf("Elément %d : ", I);
     scanf("%d", &A[I]);
    }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<N; I++)
     printf("%d ", A[I]);
 printf("\n");
 /* Recherche du maximum et du minimum */
 MIN=0;
 MAX=0;
 for (I=0; I<N; I++)
     {
      if(A[I]>A[MAX]) MAX=I;
      if(A[I]<A[MIN]) MIN=I;
     }
  /* Edition du résultat */
 printf("Position du minimum : %d\n", MIN);
 printf("Position du maximum : %d\n", MAX);
 printf("Valeur   du minimum : %d\n", A[MIN]);
 printf("Valeur   du maximum : %d\n", A[MAX]);
 return 0;
}
```

Exercice 7.11 Insérer une valeur dans un tableau trié

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50]; /* tableau donné    */
 int VAL;   /* valeur à insérer */
 int N;     /* dimension        */
 int I;     /* indice courant   */

 /* Saisie des données */
 printf("Dimension N du tableau initial (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
    {
     printf("Elément %d : ", I);
     scanf("%d", &A[I]);
    }
```

```
 printf("Elément à insérer : ");
 scanf("%d", &VAL );
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
    printf("%d ", A[I]);
 printf("\n");
 /* Déplacer les éléments plus grands que */
 /* VAL d'une position vers l'arrière.    */
 for (I=N ; (I>0)&&(A[I-1]>VAL) ; I--)
       A[I]=A[I-1];
 /* VAL est copié à la position du dernier */
 /* élément déplacé. */
 A[I]=VAL;
 /* Nouvelle dimension du tableau ! */
 N++;
  /* Edition des résultats */
 printf("Tableau résultat :\n");
 for (I=0; I<N; I++)
    printf("%d ", A[I]);
 printf("\n");
 return 0;
}
```

Exercice 7.12 Recherche d'une valeur dans un tableau

a) La recherche séquentielle

Comparer successivement les valeurs du tableau avec la valeur donnée.

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50]; /* tableau donné */
 int VAL;   /* valeur à rechercher   */
 int POS;   /* position de la valeur */
 int N;     /* dimension      */
 int I;     /* indice courant */

 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
    {
     printf("Elément %d : ", I);
     scanf("%d", &A[I]);
    }
 printf("Elément à rechercher : ");
 scanf("%d", &VAL );
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
```

```
    printf("%d ", A[I]);
 printf("\n");
 /* Recherche de la position de la valeur */
 POS = -1;
 for (I=0 ; (I<N)&&(POS==-1) ; I++)
       if (A[I]==VAL)
           POS=I;
  /* Edition du résultat */
 if (POS==-1)
    printf("La valeur recherchée ne se trouve pas "
           "dans le tableau.\n");
  else
    printf("La valeur %d se trouve à la position %d. \n",
 VAL, POS);
 return 0;
}
```

b) La recherche dichotomique ('recherche binaire', 'binary search')

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50]; /* tableau donné */
 int VAL;   /* valeur à rechercher   */
 int POS;   /* position de la valeur */
 int N;     /* dimension      */
 int I;     /* indice courant */
 int INF, MIL, SUP; /* limites du champ de recherche */

 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
    {
     printf("Elément %d : ", I);
     scanf("%d", &A[I]);
    }
 printf("Elément à rechercher : ");
 scanf("%d", &VAL );
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
    printf("%d ", A[I]);
 printf("\n");
 /* Initialisation des limites du domaine de recherche */
 INF=0;
 SUP=N-1;
 /* Recherche de la position de la valeur */
 POS=-1;
 while ((INF<=SUP) && (POS==-1))
        {
         MIL=(SUP+INF)/2;
```

```
        if (VAL < A[MIL])
              SUP=MIL-1;
        else if (VAL > A[MIL])
              INF=MIL+1;
        else
              POS=MIL;
       }

  /* Edition du résultat */
 if (POS==-1)
     printf("La valeur recherchée ne se trouve pas "
            "dans le tableau.\n");
 else
     printf("La valeur %d se trouve à la position %d. \n",
 VAL, POS);
 return 0;
}
```

Question: ***Quel est l'avantage de la recherche dichotomique?***

Dans le pire des cas d'une recherche séquentielle, il faut traverser tout le tableau avant de trouver la valeur ou avant d'être sûr qu'une valeur ne se trouve pas dans le tableau.

Lors de la recherche dichotomique, on élimine la moitié des éléments du tableau à chaque exécution de la boucle. Ainsi, la recherche se termine beaucoup plus rapidement.

La recherche dichotomique devient extrêmement avantageuse pour la recherche dans de grands tableaux (triés) : L'avantage de la recherche dichotomique par rapport à la recherche séquentielle monte alors *exponentiellement* avec la grandeur du tableau à trier.

Exemple:

Lors de la recherche dans un tableau de 1024 éléments:

- le pire des cas pour la recherche séquentielle peut entraîner 1024 exécutions de la boucle.

- le pire des cas pour la recherche dichotomique peut entraîner 10 exécutions de la boucle.

Lors de la recherche dans un tableau de 1 048 576 éléments:

- le pire des cas pour la recherche séquentielle peut entraîner 1 048 576 exécutions de la boucle.

- le pire des cas pour la recherche dichotomique peut entraîner 20 exécutions de la boucle.

Exercice 7.13 Fusion de deux tableaux triés

```
#include <stdio.h>
main()
{
 /* Déclarations */
 /* Les tableaux et leurs dimensions */
 int A[50], B[50], FUS[100];
 int N, M;
 int IA, IB, IFUS; /* indices courants */

 /* Saisie des données */
 printf("Dimension du tableau A (max.50) : ");
 scanf("%d", &N );
```

```
 printf("Entrer les éléments de A dans l'ordre croissant :\n");
 for (IA=0; IA<N; IA++)
    {
     printf("Elément A[%d] : ", IA);
     scanf("%d", &A[IA]);
    }
 printf("Dimension du tableau B (max.50) : ");
 scanf("%d", &M );
 printf("Entrer les éléments de B dans l'ordre croissant :\n");
 for (IB=0; IB<M; IB++)
    {
     printf("Elément B[%d] : ", IB);
     scanf("%d", &B[IB]);
    }
 /* Affichage des tableaux A et B */
 printf("Tableau A :\n");
 for (IA=0; IA<N; IA++)
     printf("%d ", A[IA]);
 printf("\n");
 printf("Tableau B :\n");
 for (IB=0; IB<M; IB++)
     printf("%d ", B[IB]);
 printf("\n");

/* Fusion des éléments de A et B dans FUS */
 /* de façon à ce que FUS soit aussi trié. */
 IA=0; IB=0; IFUS=0;
 while ((IA<N) && (IB<M))
        if(A[IA]<B[IB])
            {
             FUS[IFUS]=A[IA];
             IFUS++;
             IA++;
            }
        else
            {
             FUS[IFUS]=B[IB];
             IFUS++;
             IB++;
            }
 /* Si IA ou IB sont arrivés à la fin de leur tableau, */
 /* alors copier le reste de l'autre tableau.          */
 while (IA<N)
        {
         FUS[IFUS]=A[IA];
         IFUS++;
         IA++;
        }
 while (IB<M)
        {
         FUS[IFUS]=B[IB];
```

```
         IFUS++;
         IB++;
        }

 /* Edition du résultat */
 printf("Tableau FUS :\n");
 for (IFUS=0; IFUS<N+M; IFUS++)
     printf("%d ", FUS[IFUS]);
 printf("\n");
 return 0;
}
```

Exercice 7.14 Tri par sélection du maximum

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50]; /* tableau donné */
 int N;     /* dimension     */
 int I;     /* rang à partir duquel A n'est pas trié */
 int J;     /* indice courant      */
 int AIDE;  /* pour la permutation */
 int PMAX;  /* indique la position de l'élément */
            /* maximal à droite de A[I]         */

 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (J=0; J<N; J++)
    {
     printf("Elément %d : ", J);
     scanf("%d", &A[J]);
    }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (J=0; J<N; J++)
    printf("%d ", A[J]);
 printf("\n");

/* Tri du tableau par sélection directe du maximum. */
 for (I=0; I<N-1; I++)
     {
      /* Recherche du maximum à droite de A[I] */
      PMAX=I;
      for (J=I+1; J<N; J++)
          if (A[J]>A[PMAX]) PMAX=J;
      /* Echange de A[I] avec le maximum */
      AIDE=A[I];
      A[I]=A[PMAX];
      A[PMAX]=AIDE;
     }
```

```
  /* Edition du résultat */
 printf("Tableau trié :\n");
 for (J=0; J<N; J++)
     printf("%d ", A[J]);
 printf("\n");
 return 0;
}
```

Exercice 7.15 Tri par propagation (bubble sort)

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50]; /* tableau donné */
 int N;     /* dimension     */
 int I;     /* rang à partir duquel A est trié */
 int J;     /* indice courant      */
 int AIDE;  /* pour la permutation */
 int FIN;   /* position où la dernière permutation a eu lieu. */
        /* permet de ne pas trier un sous-ensemble déjà trié. */

 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (J=0; J<N; J++)
    {
     printf("Elément %d : ", J);
     scanf("%d", &A[J]);
    }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (J=0; J<N; J++)
    printf("%d ", A[J]);
 printf("\n");

 /* Tri du tableau par propagation de l'élément maximal. */
 for (I=N-1 ; I>0 ; I=FIN)
     {
      FIN=0;
      for (J=0; J<I; J++)
            if (A[J]>A[J+1])
            {
             FIN=J;
             AIDE=A[J];
             A[J]=A[J+1];
             A[J+1]=AIDE;
            }
     }

  /* Edition du résultat */
```

```
 printf("Tableau trié :\n");
 for (J=0; J<N; J++)
     printf("%d ", A[J]);
 printf("\n");
 return 0;
}
```

Exercice 7.16 Statistique des notes

```
#include <stdio.h>
main()
{
 int POINTS[50];  /* tableau des points */
 int NOTES[7];    /* tableau des notes  */
 int N;           /* nombre d'élèves    */
 int I, IN;       /* compteurs d'aide   */
 int SOM;         /* somme des points   */
 int MAX, MIN;    /* maximum, minimum de points    */
 int MAXN;        /* nombre de lignes du graphique */

 /* Saisie des données */
 printf("Entrez le nombre d'élèves (max.50) : ");
 scanf("%d", &N);
 printf("Entrez les points des élèves:\n");
 for (I=0; I<N; I++)
    {printf("Elève %d:", I+1);
     scanf("%d", &POINTS[I]);
    }
 printf("\n");

 /* Calcul et affichage du maximum et du minimum des points */
 for (MAX=0, MIN=60, I=0; I<N; I++)
    {if (POINTS[I] > MAX) MAX=POINTS[I];
     if (POINTS[I] < MIN) MIN=POINTS[I];
    }
 printf("La note maximale est %d \n", MAX);
 printf("La note minimale est %d \n", MIN);
 /* Calcul et affichage de la moyenne des points */
 for (SOM=0,I=0 ; I<N ; I++)
      SOM += POINTS[I];
 printf("La moyenne des notes est %f \n", (float)SOM/N);

 /* Etablissement du tableau NOTES */
 for (IN=0 ; IN<7 ; IN++)
      NOTES[IN] = 0;
 for (I=0; I<N; I++)
      NOTES[POINTS[I]/10]++;

 /* Recherche du maximum MAXN dans NOTES */
 for (MAXN=0,IN=0 ; IN<7 ; IN++)
      if (NOTES[IN] > MAXN)
          MAXN = NOTES[IN];
```

```
 /* Affichage du graphique de barreaux */
 /* Représentation de MAXN lignes */
 for (I=MAXN; I>0; I--)
    {
     printf("\n %2d  >", I);
     for (IN=0; IN<7; IN++)
          {
           if (NOTES[IN]>=I)
               printf(" #######");
           else
               printf("        ");
          }
    }

 /* Affichage du domaine des notes */
 printf("\n      +");
 for (IN=0; IN<7; IN++)
      printf("-------+");
 printf("\n       I 0 - 9 I 10-19 I 20-29 "
        "I 30-39 I 40-49 I 50-59 I  60   I\n");
  return 0;
}
```

****

Les chaines de caractères

Exercice 8.5

Ecrire un programme qui demande l'introduction du nom et du prénom de l'utilisateur et qui affiche alors la longueur totale du nom sans compter les espaces. Employer la fonction **strlen**.

Exemple:

```
Introduisez votre nom et votre prénom:
Mickey Mouse

Bonjour Mickey Mouse !
Votre nom est composé de 11 lettres.
```

Exercice 8.6

Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2, les compare lexicographiquement et affiche le résultat:

Exemple:

```
Introduisez la première chaîne: ABC
Introduisez la deuxième chaîne: abc
"ABC" précède "abc"
```

Exercice 8.7

Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2 et qui copie la première moitié de CH1 et la première moitié de CH2 dans une troisième chaîne CH3. Afficher le résultat.

a) Utiliser les fonctions spéciales de <*string*>.

b) Utiliser uniquement les fonctions **gets** et **puts**.

Exercice 8.8

Ecrire un programme qui lit un verbe régulier en "er" au clavier et qui en affiche la conjugaison au présent de l'indicatif de ce verbe. Contrôlez s'il s'agit bien d'un verbe en "er" avant de conjuguer. Utiliser les fonctions **gets**, **puts**, **strcat** et **strlen**.

Exemple:

```
Verbe : fêter
je fête
tu fêtes
il fête
nous fêtons
vous fêtez
ils fêtent
```

Exercice 8.12

Ecrire un programme qui lit un nombre entre 1 et 7 et qui affiche le nom du jour de la semaine correspondant:

`"lundi"`	pour 1
`"mardi"`	pour 2
. . .	. . .
`"dimanche"`	pour 7

Utiliser le premier élément du tableau pour mémoriser un petit message d'erreur.

Exercice 8.13

Ecrire un programme qui lit 5 mots, séparés par des espaces et qui les affiche ensuite dans une ligne, mais dans l'ordre inverse. Les mots sont mémorisés dans un tableau de chaînes de caractères.

Exemple

```
voici une petite phrase !
! phrase petite une voici
```

Exercice 8.14

Refaire l'exercice 8.8 (Conjugaison des verbes réguliers en "er") en utilisant deux tableaux de chaînes de caractères:

SUJ	pour les sujets
TERM	pour les terminaisons

Employez les fonctions **printf**, **scanf**, **strlen**.

Sauf indication contraire, les exercices suivants sont à résoudre ***sans utiliser les fonctions spéciales des bibliothèques <string>, <stdlib> ou <ctype>***. Ils servent à comprendre et à suivre le raisonnement de ces fonctions.

Exercice 8.15

Ecrire un programme qui lit deux chaînes de caractères, et qui indique leur précédence lexicographique dans le code de caractères de la machine (ici: code ASCII). Testez votre programme à l'aide des exemples du chapitre 8.5.

Exercice 8.16

Ecrire un programme qui lit une chaîne de caractères CH et qui convertit toutes les majuscules dans des minuscules et vice-versa.

Le résultat sera mémorisé dans la même variable CH et affiché après la conversion.

Exercice 8.17

Ecrire une procédure qui lit une chaîne de caractères et l'interprète comme un entier positif dans la base ***décimale***. Pour la conversion, utiliser les fonctions de <*ctype*> et la précédence alphabétique des caractères de '0' à '9'. Mémoriser le résultat dans une variable du type **long**. La conversion s'arrête à la rencontre du premier caractère qui ne représente pas de chiffre décimal. Utiliser un indicateur logique OK qui précise si la chaîne représente correctement une valeur entière et positive.

Exercice 8.18

Ecrire une procédure qui lit une chaîne de caractères et l'interprète comme un entier positif dans la base ***hexadécimale***. Pour la conversion, utiliser les fonctions de <*ctype*> et la précédence alphabétique des caractères. La conversion ignore les caractères qui ne représentent pas de chiffre hexadécimal et s'arrête à la fin de la chaîne de caractères. Le résultat sera mémorisé dans une variable du type **long** et affiché dans les bases hexadécimale **et** décimale.

Exercice 8.19

En se basant sur l'exercice 8.17, écrire un programme qui lit une chaîne de caractères et l'interprète comme un nombre rationnel positif ou négatif introduit en *notation décimale*. Mémoriser le résultat dans une variable du type **double**. Si le nombre a été introduit correctement, la valeur du résultat sera affichée, sinon le programme affichera un message d'erreur.

Méthode:

Utiliser une variable SIG pour mémoriser le signe de la valeur. Convertir tous les caractères numériques (avant et derrière le point décimal) en une valeur entière N. Compter les décimales

(c.-à-d.: les positions derrière le point décimal) à l'aide d'une variable DEC et calculer la valeur rationnelle comme suit:

N = N * SIG / pow(10, DEC)

Exemples:

```
-1234.234    -1234.23400
-123 45      Erreur!
123.23.      Erreur!
+00123.0123  123.012300
```

Exercice 8.20

En se basant sur l'exercice 8.19, écrire un programme qui lit une chaîne de caractères et l'interprète comme un nombre rationnel positif ou négatif introduit en *notation exponentielle*. Mémoriser le résultat dans une variable du type **double**. Si le nombre a été introduit correctement, la valeur du résultat sera affichée, sinon le programme affichera un message d'erreur.

Méthode:

Utiliser une variable SIGE pour mémoriser le signe de l'exposant. Utiliser une variable EXP pour la valeur de l'exposant. Calculer la valeur de l'exposant à l'aide de SIGE, DEC et EXP. Calculer ensuite la valeur exacte de N à l'aide d'une formule analogue à celle de l'exercice ci-dessus.

Exemples:

```
-1234.234    -1234.234000
-1234. 234   Erreur!
123E+02      123400.000000
123E-02      1.230000
123.4e       123.400000
-12.1234e02  -1212.340000
123.4e3.4    Erreur!
12.12E1      121.200000
12.12 E1     Erreur!
```

Exercice 8.21

Ecrire un programme qui supprime la première occurrence d'une chaîne de caractères OBJ dans une chaîne de caractères SUJ.

Exemples:

```
PHON ALPHONSE     ALSE
EI   PIERRE       PIERRE
T    TOTALEMENT   OTALEMENT
     HELLO        HELLO
```

Exercice 8.22

Ecrire un programme qui remplace la première occurrence d'une chaîne de caractères CH1 par la chaîne CH2 dans une chaîne de caractères SUJ. Utiliser une chaîne de sauvegarde FIN pendant le remplacement.

Exemples:

PHON	OY	ALPHONSE	ALOYSE
IE	EI	PIERRE	PEIRRE
IE	ARTE	PIERRE	PARTERRE
EI	IE	PIERRE	PIERRE
TOT	FIN	TOTALEMENT	FINALEMENT
	TTT	HELLO	HELLO

Exercice 8.23

Ecrire un programme qui remplace toutes les occurrences d'une chaîne de caractères CH1 par la chaîne CH2 dans une chaîne de caractères SUJ. Utiliser une chaîne de sauvegarde FIN pendant le remplacement.

Exemples:

PHON	OY	ALPHONSE	ALOYSE
AN	ONT	BANANE	BONTONTE
T	Y	TOTALEMENT	YOYALEMENY
	TTT	HELLO	HELLO
L		HELLO	HEO

Correction

Exercice 8.5

```
#include <stdio.h>
#include <string.h>
 main()
{
  char NOM[40], PRENOM[40];
  printf("Introduisez votre nom et votre prénom: \n");
  scanf("%s %s", NOM, PRENOM);
  printf("\nBonjour %s %s !\n", NOM, PRENOM);
  printf("Votre nom est composé de %d lettres.\n",
                                  strlen(NOM) + strlen(PRENOM));
 /* ou bien
  printf("Votre nom est composé de %d lettres.\n",
                                  strlen(strcat(NOM,PRENOM)));
 */
 return 0;
}
```

Exercice 8.6

```
#include <stdlib.h>
#include <string.h>
 main()
{
 /* Déclarations */
 char CH1[200], CH2[200]; /* chaînes entrées */
 int RES;  /* résultat de la fonction strcmp */

 printf("Introduisez la première chaîne de caractères : ");
 gets(CH1);
 printf("Introduisez la deuxième chaîne de caractères : ");
 gets(CH2);

 /* Comparaison et affichage du résultat */
 RES = strcmp(CH1,CH2);
 if (RES<0)
     printf("\"%s\" précède \"%s\"\n",CH1 ,CH2);
 else if (RES>0)
     printf("\"%s\" précède \"%s\"\n",CH2 ,CH1);
 else
     printf("\"%s\" est égal à \"%s\"\n",CH1, CH2);
  return 0;
}
```

Exercice 8.7

a) Utiliser les fonctions spéciales de <*string*>.

```
#include <stdio.h>
#include <string.h>
 main()
{
 /* Déclarations */
 char CH1[100], CH2[100]; /* chaînes données */
 char CH3[100]="";        /* chaîne résultat */

 /* Saisie des données */
 printf("Introduisez la première chaîne de caractères : ");
 gets(CH1);
 printf("Introduisez la deuxième chaîne de caractères : ");
 gets(CH2);


 /* Traitements */
 strncpy(CH3, CH1, strlen(CH1)/2);
 strncat(CH3, CH2, strlen(CH2)/2);
  /* Affichage du résultat */
 printf("Un demi \"%s\" plus un demi \"%s\" donne \"%s\"\n",
                                              CH1, CH2, CH3);
 return 0;
```

```
}
```

b) Utiliser uniquement les fonctions **gets** et **puts**.

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char CH1[100], CH2[100]; /* chaînes données  */
 char CH3[100]="";        /* chaîne résultat  */
 int L1,L2; /* longueurs de CH1 et CH2        */
 int I;     /* indice courant dans CH1 et CH2 */
 int J;     /* indice courant dans CH3        */

 /* Saisie des données */
 puts("Introduisez la première chaîne de caractères : ");
 gets(CH1);
 puts("Introduisez la deuxième chaîne de caractères : ");
 gets(CH2);

 /* Détermination les longueurs de CH1 et CH2 */
 for (L1=0; CH1[L1]; L1++) ;
 for (L2=0; CH2[L2]; L2++) ;
 /* Copier la première moitié de CH1 vers CH3 */
 for (I=0 ; I<(L1/2) ; I++)
     CH3[I]=CH1[I];
 /* Copier la première moitié de CH2 vers CH3 */
 J=I;
 for (I=0 ; I<(L2/2) ; I++)
    {
     CH3[J]=CH2[I];
     J++;
    }
 /* Terminer la chaîne CH3 */
 CH3[J]='\0';

 /* Affichage du résultat */
 puts("Chaîne résultat : ");
 puts(CH3);
 return 0;
}
```

Exercice 8.8

```
#include <stdio.h>
#include <string.h>
 main()
{
 /* Déclarations */
 char VERB[20]; /* chaîne contenant le verbe */
 char AFFI[30]; /* chaîne pour l'affichage   */
 int L;         /* longueur de la chaîne     */

 /* Saisie des données */
```

```
 printf("Verbe : ");
 gets(VERB);

 /* Contrôler s'il s'agit d'un verbe en 'er' */
 L=strlen(VERB);
 if ((VERB[L-2]!='e') || (VERB[L-1]!='r'))
      puts("\aCe n'est pas un verbe du premier groupe.!");
 else
     {
      /* Couper la terminaison 'er'. */
      VERB[L-2]='\0';
      /* Conjuguer ... */
      AFFI[0]='\0';
      strcat(AFFI, "je ");
      strcat(AFFI, VERB);
      strcat(AFFI, "e");
      puts(AFFI);

        . . .

      AFFI[0]='\0';
      strcat(AFFI, "ils ");
      strcat(AFFI, VERB);
      strcat(AFFI, "ent");
      puts(AFFI);
     }
  return 0;
}
```

Exercice 8.10

```
#include <stdio.h>
#include <stdlib.h>
 main()
{
 long N;
 char STR[200];
 do
    {
     puts("Entrez un nombre :");
     scanf("%ld",&N);
     printf("Entrée  = %ld\n", N);
     printf("base 2  = %s\n", ltoa(N, STR, 2));
     printf("base 8  = %s\n", ltoa(N, STR, 8));
     printf("base 16 = %s\n", ltoa(N, STR, 16));
    }
 while(N);
 return 0;
}
```

Exercice 8.11

```
#include <stdio.h>
#include <string.h>
main()
{
 /* Déclarations */
 char MOT[10][50]; /* tableau de 10 mots à trier */
  char AIDE[50];  /* chaîne d'aide pour la permutation */
 int I;    /* rang à partir duquel MOT n'est pas trié */
 int J;    /* indice courant */
 int PMOT; /* indique la position du prochain mot */
           /* dans la suite lexicographique.       */

 /* Saisie des données */
 for (J=0; J<10; J++)
   {
    printf("Mot %d : ", J);
    gets(MOT[J]); /* ou :  scanf ("%s\n", MOT[J]); */
    }

 /* Tri du tableau par sélection directe du     */
 /* prochain mot dans la suite lexicographique. */
 for (I=0; I<9; I++)
    {
     /* Recherche du prochain mot à droite de A[I] */
     PMOT=I;
     for (J=I+1; J<10; J++)
        if (strcmp(MOT[J], MOT[PMOT]) < 0)
            PMOT=J;
     /* Echange des mots à l'aide de strcpy */
     strcpy(AIDE, MOT[I]);
     strcpy(MOT[I], MOT[PMOT]);
     strcpy(MOT[PMOT], AIDE);
    }

 /* Edition du résultat */
 printf("Tableau trié lexicographiquement :\n");
 for (J=0; J<10; J++)
     puts(MOT[J]);  /* ou :  printf("%s\n",MOT[J]); */
 printf("\n");
 return 0;
}
```

Exercice 8.12

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int N;  /* nombre entré */
```

```
 char JOUR[8][9] = {"\aErreur!", "lundi", "mardi", "mercredi",
                    "jeudi", "vendredi", "samedi","dimanche"};
  /* Saisie du nombre */
 printf("Entrez un nombre entre 1 et 7 : ");
 scanf("%d", &N);
 /* Affichage du résultat - pour perfectionnistes */
 if (N>0 && N<8)
     printf("Le %de%c jour de la semaine est %s.\n",
                                    N, (N==1)?'r':' ', JOUR[N]);
 else
     puts(JOUR[0]);
  return 0;
}
```

Exercice 8.13

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char MOT[5][50]; /* tableau pour les 5 mots */
 int I;           /* indice courant */
 /* Saisie des mots */
 printf("Entrez 5 mots, séparés par des espaces :\n");
 /* Après le retour à la ligne, scanf lit */
 /* les 5 données à la fois. */
 for (I=0; I<5; I++)
     scanf("%s", MOT[I]);
 /* Affichage du résultat */
 for (I=4; I>=0; I--)
    printf("%s ", MOT[I]);
 printf("\n");
  return 0;
}
```

Exercice 8.14

```
#include <stdio.h>
#include <string.h>
main()
{
 /* Déclarations */
 /* Sujets et terminaisons */
 char SUJ[6][5]  = {"je","tu","il","nous","vous","ils"};
 char TERM[6][5] = {"e","es","e","ons","ez","ent"};
  char VERB[20]; /* chaîne contenant le verbe */
 int L;         /* longueur de la chaîne */
 int I;         /* indice courant */

 /* Saisie des données */
 printf("Verbe : ");
```

```
 scanf("%s", VERB);

 /* Contrôler s'il s'agit d'un verbe en 'er' */
 L=strlen(VERB);
 if ((VERB[L-2] != 'e') || (VERB[L-1] != 'r'))
  printf("\"%s\" n'est pas un verbe du premier groupe.\n",VERB);
 else
     {
      /* Couper la terminaison 'er'. */
      VERB[L-2]='\0';
      /* Conjuguer ... */
      for (I=0; I<6; I++)
          printf("%s %s%s\n",SUJ[I], VERB, TERM[I]);
      }
 return 0;
}
```

Exercice 8.15

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char CH1[50], CH2[50]; /* chaînes à comparer */
 int I;                 /* indice courant     */

 /* Saisie des données */
 printf("Entrez la première chaîne à comparer : ");
 gets(CH1);
 printf("Entrez la deuxième chaîne à comparer : ");
 gets(CH2);

 /* Chercher la première position où */
 /* CH1 et CH2 se distinguent. */
 for (I=0; (CH1[I]==CH2[I]) && CH1[I] && CH2[I]; I++)
     ;
 /* Comparer le premier élément qui */
 /* distingue CH1 et CH2. */
 if (CH1[I]==CH2[I])
     printf("\"%s\" est égal à \"%s\"\n", CH1, CH2);
 else if (CH1[I]<CH2[I])
     printf("\"%s\" précède \"%s\"\n", CH1, CH2);
 else
     printf("\"%s\" précède \"%s\"\n", CH2, CH1);
  return 0;
}
```

Exercice 8.16

```
#include <stdio.h>
main()
{
```

```
 /* Déclarations */
 char CH[100]; /* chaîne à convertir */
 int I;        /* indice courant     */

 /* Saisie de la chaîne */
 printf("Entrez la chaîne à convertir : ");
 gets(CH);
 /* Conversion de la chaîne */
 for (I=0; CH[I]; I++)
    {
     if (CH[I]>='A' && CH[I]<='Z')
                 CH[I] = CH[I]-'A'+'a';
     else if (CH[I]>='a' && CH[I]<='z')
                 CH[I] = CH[I]-'a'+'A';
    }
  /* Affichage de la chaîne convertie */
 printf("Chaîne convertie : %s\n", CH);
 return 0;
}
```

Exercice 8.17

```
#include <stdio.h>
#include <ctype.h>
 main()
{
 /* Déclarations */
 char CH[100]; /* chaîne numérique à convertir */
 long N; /* résultat numérique */
 int I;  /* indice courant */
 int OK; /* indicateur logique précisant si la */
         /* chaîne a été convertie avec succès */

 /* Saisie de la chaîne */
 printf("Entrez un nombre entier et positif : ");
 gets(CH);
 /* Conversion de la chaîne */
 OK=1;
 N=0;
 for (I=0; OK && CH[I]; I++)
     if (isdigit(CH[I]))
        N = N*10 + (CH[I]-'0');
     else
        OK=0;

 /* Affichage de la chaîne convertie */
 if (OK)
     printf("Valeur numérique : %ld\n", N);
 else
     printf("\a\"%s\" ne représente pas correctement "
           "un entier et positif.\n", CH);
```

```
 return 0;
}
```

Exercice 8.18

```
#include <stdio.h>
#include <ctype.h>
 main()
{
 /* Déclarations */
 char CH[100]; /* chaîne numérique à convertir */
 long N; /* résultat numérique */
 int I;  /* indice courant */
 int OK; /* indicateur logique précisant si la */
         /* chaîne a été convertie avec succès */

 /* Saisie de la chaîne */
 printf("Entrez un nombre hexadécimal entier et positif : ");
 gets(CH);
 /* Conversion de la chaîne */
 OK=1;
 N=0;
 for (I=0; OK && CH[I]; I++)
     if (isxdigit(CH[I]))
       {
        CH[I] = toupper(CH[I]);
        if (isdigit(CH[I]))
           N = N*16 + (CH[I]-'0');
        else
           N = N*16 + 10 + (CH[I]-'A');
       }
     else
        OK=0;

 /* Affichage de la chaîne convertie */
 if (OK)
   {
    printf("Valeur numérique hexadécimale : %lX\n", N);
    printf("Valeur numérique décimale     : %ld\n", N);
   }
 else
    printf("\a\"%s\" n'est pas une valeur "
           "hexadécimale correcte.\n", CH);
 return 0;
}
```

Exercice 8.19

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>
```

```
main()
{
 /* Déclarations */
 char CH[100]; /* chaîne numérique à convertir */
 double N; /* résultat numérique               */
 int I;    /* indice courant                   */
 int SIG;  /* signe de la valeur rationnelle   */
 int DEC;  /* nombre de décimales              */
 int OK;   /* indicateur logique précisant si la */
           /* chaîne a été convertie avec succès */

 /* Saisie de la chaîne */
 printf("Entrez un nombre rationnel : ");
 gets(CH);

 /* Conversion de la chaîne :    */
 /* Initialisation des variables */
  OK=1;
 N=0.0;
 I=0;
 SIG=1;
 /* Traitement du signe */
  if (CH[I]=='-') SIG=-1;
 if (CH[I]=='-' || CH[I]=='+')
     I++;
 /* Positions devant le point décimal */
  for ( ; isdigit(CH[I]); I++)
     N = N*10.0 + (CH[I]-'0');
 /* Traitement du point décimal */
  if (CH[I]=='.')
     I++;
 else if (CH[I])
     OK=0;

 /* Traitement et comptage des décimales */
  for (DEC=0; isdigit(CH[I]); I++, DEC++)
     N = N*10.0 + (CH[I]-'0');
 if (CH[I]) OK=0;
 /* Calcul de la valeur à partir du signe et */
 /* du nombre de décimales. */
  N = SIG*N/pow(10,DEC);
 /* Affichage de la chaîne convertie */
 if (OK)
    printf("Valeur numérique : %f\n", N);
 else
    printf("\a\"%s\" n'est pas une valeur "
           "rationnelle correcte.\n", CH);
 return 0;
}
```

Exercice 8.20

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>
main()
{/* Déclarations */
 char CH[100]; /* chaîne numérique à convertir */
 double N; /* résultat numérique */
 int I;    /* indice courant */
 int SIG;  /* signe de la valeur rationnelle */
 int DEC;  /* nombre de décimales  */
 int SIGE; /* signe de l'exposant  */
 int EXP;  /* valeur de l'exposant */
 int OK;   /* indicateur logique précisant si la */
           /* chaîne a été convertie avec succès */

 /* Saisie de la chaîne */
 printf("Entrez un nombre rationnel : ");
 gets(CH);

 /* Conversion de la chaîne */
 /* Initialisation des variables */
  OK=1;
 N=0.0;
 I=0;
 SIG=1;
 SIGE=1;
 /* Traitement du signe */
  if (CH[I]=='-') SIG=-1;
 if (CH[I]=='-' || CH[I]=='+') I++;
 /* Positions devant le point décimal */
  for ( ; isdigit(CH[I]); I++)
     N = N*10.0 + (CH[I]-'0');
 /* Traitement du point décimal */
  if (CH[I]=='.')
     I++;
 /* Traitement et comptage des décimales */
  for (DEC=0; isdigit(CH[I]); I++, DEC++)
     N = N*10.0 + (CH[I]-'0');
 /* Traitement de la marque exponentielle */
  if (CH[I]=='e'||CH[I]=='E')
     I++;
 else if (CH[I])
     OK=0;
 /* Traitement du signe de l'exposant */
  if (CH[I]=='-') SIGE=-1;
 if (CH[I]=='-' || CH[I]=='+') I++;
 /* Traitement de la valeur de l'exposant */
  for (EXP=0; isdigit(CH[I]); I++)
     EXP = EXP*10 + (CH[I]-'0');
```

```
 if (CH[I]) OK=0;
 /* Calcul de l'exposant à partir du signe  */
 /* SIGE, de la valeur de l'exposant EXP et */
 /* du nombre de positions rationnelles DEC */
  EXP = SIGE*EXP - DEC;
 /* Calcul de la valeur à partir du signe et */
 /* de l'exposant. */
  N = SIG*N*pow(10,EXP);

 /* Affichage de la chaîne convertie */
 if (OK)
    printf("Valeur numérique : %f\n", N);
 else
    printf("\a\"%s\" n'est pas une valeur "
           "rationnelle correcte.\n", CH);
 return 0;
}
```

Exercice 8.21

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char SUJ[100]; /* chaîne à transformer        */
 char OBJ[100]; /* chaîne à supprimer dans SUJ */
 int I;         /* indice courant dans SUJ     */
 int J;         /* indice courant dans OBJ     */
 int TROUVE;    /* indicateur logique qui précise */
                /* si la chaîne OBJ a été trouvée */

 /* Saisie des données */
 printf("Introduisez la chaîne à supprimer   : ");
 gets(OBJ);
 printf("Introduisez la chaîne à transformer : ");
 gets(SUJ);
 /* Recherche de OBJ dans SUJ */
 TROUVE=0;
 for (I=0; SUJ[I] && !TROUVE; I++)
    /* Si la première lettre est identique, */
    if (SUJ[I]==OBJ[0])
        {
         /* alors comparer le reste de la chaîne */
         for (J=1; OBJ[J] && (OBJ[J]==SUJ[I+J]); J++)
            ;
         if (OBJ[J]=='\0') TROUVE=1;
        }
 /* Si la position de départ de OBJ dans SUJ a été trouvée */
 /* alors déplacer le reste de SUJ à cette position. */
 if (TROUVE)
    {
     I--;
```

```
     /* Maintenant  I indique la position de OBJ */
     /* dans SUJ et J indique la longueur de OBJ */
     for (; SUJ[I+J]; I++)
          SUJ[I]=SUJ[I+J];
     SUJ[I]='\0';
    }
  /* Affichage du résultat */
 printf("Chaîne résultat : \"%s\"\n", SUJ);
 return 0;
}
```

Exercice 8.22

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char SUJ[100]; /* chaîne à transformer */
 char CH1[100]; /* chaîne à rechercher  */
 char CH2[100]; /* chaîne de remplacement */
 char FIN[100]; /* chaîne de sauvegarde pour */
                /* la fin de SUJ. */
 int I;      /* indice courant dans SUJ */
 int J;      /* indice courant dans CH1 et CH2 */
 int K;      /* indice d'aide pour les copies  */
 int TROUVE; /* indicateur logique qui précise */
             /* si la chaîne OBJ a été trouvée */

 /* Saisie des données */
 printf("Introduisez la chaîne à rechercher  CH1 : ");
 gets(CH1);
 printf("Introduisez la chaîne à remplacer   CH2 : ");
 gets(CH2);
 printf("Introduisez la chaîne à transformer SUJ : ");
 gets(SUJ);

 /* Recherche de CH1 dans SUJ */
 TROUVE=0;
 for (I=0; SUJ[I] && !TROUVE; I++)
    if (SUJ[I]==CH1[0])
        {
         for (J=1; CH1[J] && (CH1[J]==SUJ[I+J]); J++)
             ;
         if (CH1[J]=='\0') TROUVE=1;
        }


 /* Si CH1 a été trouvée dans SUJ alors sauvegarder la fin */
 /* de SUJ dans FIN, copier ensuite CH2 et FIN dans SUJ.   */
 if (TROUVE)
    {
     I--;
```

```
     /* Maintenant  I indique la position de CH1 */
     /* dans SUJ et J indique la longueur de CH1 */
     /* Sauvegarder la fin de SUJ dans FIN */
     for (K=0; SUJ[K+I+J]; K++)
          FIN[K]=SUJ[K+I+J];
     FIN[K]='\0';
     /* Copier CH2 dans SUJ */
     for (K=0; CH2[K]; K++,I++)
          SUJ[I]=CH2[K];
     /* Recopier FIN dans SUJ */
     for (K=0; FIN[K]; K++,I++)
          SUJ[I]=FIN[K];
     /* Terminer la chaîne SUJ */
     SUJ[I]='\0';
    }

 /* Affichage du résultat */
 printf("Chaîne résultat : \"%s\"\n", SUJ);
 return 0;
}
```

Exercice 8.23

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char SUJ[100]; /* chaîne à  transformer */
 char CH1[100]; /* chaîne à  rechercher  */
 char CH2[100]; /* chaîne de remplacement */
 char FIN[100]; /* chaîne de sauvegarde pour */
                /* la fin de SUJ. */
 int I;      /* indice courant dans SUJ */
 int J;      /* indice courant dans CH1 et CH2 */
 int K;      /* indice d'aide pour les copies  */


 /* Saisie des données */
 printf("Introduisez la chaîne à rechercher  CH1 : ");
 gets(CH1);
 printf("Introduisez la chaîne à remplacer   CH2 : ");
 gets(CH2);
 printf("Introduisez la chaîne à transformer SUJ : ");
 gets(SUJ);

 /* Recherche de CH1 dans SUJ */
 for (I=0; SUJ[I]; I++)
    if (SUJ[I]==CH1[0])
       {
        for (J=1; CH1[J] && (CH1[J]==SUJ[I+J]); J++)
            ;
        if (CH1[J]=='\0') /* TROUVE ! */
```

```
            {
             /* Maintenant  I indique la position de CH1 */
             /* dans SUJ et J indique la longueur de CH1 */
             /* Sauvegarder la fin de SUJ dans FIN */
             for (K=0; SUJ[K+I+J]; K++)
                FIN[K]=SUJ[K+I+J];
             FIN[K]='\0';
             /* Copier CH2 dans SUJ et déplacer */
             /* I derrière la copie de CH2. */
             for (K=0; CH2[K]; K++,I++)
                 SUJ[I]=CH2[K];
             /* Recopier FIN dans SUJ */
             for (K=0; FIN[K]; K++)
                 SUJ[I+K]=FIN[K];
             /* Terminer la chaîne SUJ */
             SUJ[I+K]='\0';
             I--; /* réajustement de l'indice I */
            }
        }

 /* Affichage du résultat */
 printf("Chaîne résultat : \"%s\"\n", SUJ);
 return 0;
}
```

****

LES POINTEURS

Exercice 9.7

Ecrire un programme qui lit deux tableaux d'entiers A et B et leurs dimensions N et M au clavier et qui ajoute les éléments de B à la fin de A. Utiliser deux pointeurs PA et PB pour le transfer et afficher le tableau résultant A.

Exercice 9.8

Ecrire de deux façons différentes, un programme qui vérifie sans utiliser une fonction de <*string*>, si une chaîne CH introduite au clavier est un palindrome:

a) en utilisant uniquement le formalisme tableau

b) en utilisant des pointeurs au lieu des indices numériques

Rappel: Un palindrome est un mot qui reste le même qu'on le lise de gauche à droite ou de droite à gauche:

Exemples: `PIERRE` ==> n'est pas un palindrome

`OTTO` ==> est un palindrome

`23432` ==> est un palindrome

Exercice 9.9

Ecrire un programme qui lit une chaîne de caractères CH et détermine la longueur de la chaîne à l'aide d'un pointeur P. Le programme n'utilisera pas de variables numériques.

Exercice 9.10

Ecrire un programme qui lit une chaîne de caractères CH et détermine le nombre de mots contenus dans la chaîne. Utiliser un pointeur P, une variable logique, la fonction **isspace** et une variable numérique N qui contiendra le nombre des mots.

Exercice 9.11

Ecrire un programme qui lit une chaîne de caractères CH au clavier et qui compte les occurrences des lettres de l'alphabet en ne distinguant pas les majuscules et les minuscules. Utiliser un tableau ABC de dimension 26 pour mémoriser le résultat et un pointeur PCH pour parcourir la chaîne CH et un pointeur PABC pour parcourir ABC. Afficher seulement le nombre des lettres qui apparaissent au mois une fois dans le texte.

Exemple:

```
Entrez un ligne de texte (max. 100 caractères) :
Jeanne
La chaîne "Jeanne" contient :
1   fois la lettre 'A'
2   fois la lettre 'E'
1   fois la lettre 'J'
3   fois la lettre 'N'
```

Exercice 9.12

Ecrire un programme qui lit un caractère C et une chaîne de caractères CH au clavier. Ensuite toutes les occurrences de C dans CH seront éliminées. Le reste des caractères dans CH sera tassé à l'aide d'un pointeur et de la fonction **strcpy**.

Exercice 9.13

Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2 au clavier et élimine toutes les lettres de CH1 qui apparaissent aussi dans CH2. Utiliser deux pointeurs P1 et P2, une variable logique TROUVE et la fonction **strcpy**.

Exemples: Bonjour Bravo ==> njou

Bonjour bravo ==> Bnjou

abacab aa ==> bcab

Exercice 9.14

Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2 au clavier et supprime la première occurrence de CH2 dans CH1. Utiliser uniquement des pointeurs, une variable logique TROUVE et la fonction **strcpy**.

Exemples: Alphonse phon ==> Alse

totalement t ==> otalement

abacab aa ==> abacab

Correction des ex9.1 jusqu'à 9.14

Exercice 9.7

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[100], B[50]; /* tableaux */
 int N, M;    /* dimensions des tableaux */
 int *PA,*PB; /* pointeurs d'aide        */

  /* Saisie des données */
 printf("Dimension du tableau A (max.50) : ");
 scanf("%d", &N );
 for (PA=A; PA<A+N; PA++)
     {
      printf("Elément %d : ", PA-A);
      scanf("%d", PA);
     }
  printf("Dimension du tableau B (max.50) : ");
 scanf("%d", &M );
 for (PB=B; PB<B+M; PB++)
     {
      printf("Elément %d : ", PB-B);
      scanf("%d", PB);
     }
  /* Affichage des tableaux */
```

```
 printf("Tableau donné A :\n");
 for (PA=A; PA<A+N; PA++)
      printf("%d ", *PA);
  printf("\n");
 printf("Tableau donné B :\n");
 for (PB=B; PB<B+M; PB++)
      printf("%d ", *PB);
  printf("\n");
 /* Copier B à la fin de A */
 for (PA=A+N,PB=B ; PB<B+M ; PA++,PB++)
      *PA = *PB;
 /* Nouvelle dimension de A */
 N += M;
  /* Edition du résultat */
 printf("Tableau résultat A :\n");
 for (PA=A; PA<A+N; PA++)
      printf("%d ", *PA);
 printf("\n");
 return 0;
}
```

Exercice 9.8

a) en utilisant uniquement le formalisme tableau

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char CH[101]; /* chaîne donnée    */
 int I,J;      /* indices courants */
 int PALI;     /* indicateur logique:          */
               /* vrai si CH est un palindrome */

  /* Saisie des données */
 printf("Entrez une ligne de texte (max.100 caractères) :\n");
 gets(CH);
 /* Placer J sur la dernière lettre de la chaîne */
 for(J=0; CH[J]; J++)
     ;
 J--;
 /* Contrôler si CH est un palindrome */
 PALI=1;
 for (I=0 ; PALI && I<J ; I++,J--)
      if (CH[I] != CH[J])
          PALI=0;

  /* Affichage du résultat */
 if (PALI)
    printf("La chaîne \"%s\" est un palindrome.\n", CH);
 else
    printf("La chaîne \"%s\" n'est pas un palindrome.\n", CH);
 return 0;
```

```
}
```

b) en utilisant des pointeurs au lieu des indices numériques :

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char CH[101];  /* chaîne donnée    */
 char *P1,*P2;  /* pointeurs d'aide */
  int PALI;  /* indicateur logique:          */
            /* vrai si CH est un palindrome */

  /* Saisie des données */
 printf("Entrez une ligne de texte (max.100 caractères) :\n");
 gets(CH);
 /* Placer P2 sur la dernière lettre de la chaîne */
 for (P2=CH; *P2; P2++)
      ;
 P2--;
 /* Contrôler si CH est un palindrome */
 PALI=1;
 for (P1=CH ; PALI && P1<P2 ; P1++,P2--)
      if (*P1 != *P2) PALI=0;
  /* Affichage du résultat */
 if (PALI)
    printf("La chaîne \"%s\" est un palindrome.\n", CH);
 else
    printf("La chaîne \"%s\" n'est pas un palindrome.\n", CH);
 return 0;
}
```

Exercice 9.9

```
#include <stdio.h>

main()
{
 /* Déclarations */
 char CH[101];  /* chaîne donnée   */
 char *P;       /* pointeur d'aide */

  /* Saisie des données */
 printf("Entrez une ligne de texte (max.100 caractères) :\n");
 gets(CH);
 /* Placer P à la fin de la chaîne */
 for (P=CH; *P; P++)
      ;
  /* Affichage du résultat */
 printf("La chaîne \"%s\" est formée de %d caractères.\n",
                                                    CH, P-CH);
 return 0;
}
```

Exercice 9.10

```
#include <stdio.h>
#include <ctype.h>
 main()
{
 /* Déclarations */
 char CH[101]; /* chaîne donnée */
 char *P;      /* pointeur d'aide */
 int N;        /* nombre des mots */
 int DANS_MOT; /* indicateur logique: */
               /* vrai si P pointe à l'intérieur un mot */

 /* Saisie des données */
 printf("Entrez une ligne de texte (max.100 caractères) :\n");
 gets(CH);
 /* Compter les mots */
 N=0;
 DANS_MOT=0;
 for (P=CH; *P; P++)
     if (isspace(*P))
         DANS_MOT=0;
     else if (!DANS_MOT)
         {
          DANS_MOT=1;
          N++;
         }
  /* Affichage du résultat (pour perfectionnistes) */
 printf("La chaîne \"%s\" \nest formée de %d mot%c.\n",
                                        CH, N, (N==1)?' ':'s');
 return 0;
}
```

Exercice 9.11

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char CH[101]; /* chaîne donnée */
 char *PCH;    /* pointeur d'aide dans CH  */
 int ABC[26];  /* compteurs des différents caractères */
  int *PABC;    /* pointeur d'aide dans ABC */

 /* Saisie des données */
 printf("Entrez une ligne de texte (max.100 caractères) :\n");
 gets(CH);
 /* Initialiser le tableau ABC */
 for (PABC=ABC; PABC<ABC+26; PABC++)
     *PABC=0;
 /* Compter les lettres */
```

```
 for (PCH=CH; *PCH; PCH++)
     {
      if (*PCH>='A' && *PCH<='Z')
         (*(ABC+(*PCH-'A')))++; /* Attention aux parenthèses! */
      if (*PCH>='a' && *PCH<='z')
         (*(ABC+(*PCH-'a')))++;
     }
 /* Affichage des résultats */
 /* (PABC-ABC) est le numéro de la lettre de l'alphabet. */
 printf("La chaîne \"%s\" contient :\n", CH);
 for (PABC=ABC; PABC<ABC+26; PABC++)
     if (*PABC)
        printf(" %d\tfois la lettre '%c' \n",
                                          *PABC, 'A'+(PABC-ABC));
  return 0;
}
```

Exercice 9.11

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char CH[101]; /* chaîne donnée */
 char *PCH;    /* pointeur d'aide dans CH  */
 int ABC[26];  /* compteurs des différents caractères */
  int *PABC;    /* pointeur d'aide dans ABC */

 /* Saisie des données */
 printf("Entrez une ligne de texte (max.100 caractères) :\n");
 gets(CH);
 /* Initialiser le tableau ABC */
 for (PABC=ABC; PABC<ABC+26; PABC++)
     *PABC=0;
 /* Compter les lettres */
 for (PCH=CH; *PCH; PCH++)
     {
      if (*PCH>='A' && *PCH<='Z')
         (*(ABC+(*PCH-'A')))++; /* Attention aux parenthèses! */
      if (*PCH>='a' && *PCH<='z')
         (*(ABC+(*PCH-'a')))++;
     }
 /* Affichage des résultats */
 /* (PABC-ABC) est le numéro de la lettre de l'alphabet. */
 printf("La chaîne \"%s\" contient :\n", CH);
 for (PABC=ABC; PABC<ABC+26; PABC++)
     if (*PABC)
        printf(" %d\tfois la lettre '%c' \n",
                                          *PABC, 'A'+(PABC-ABC));
  return 0;
}
```

Exercice 9.13

```
#include <stdio.h>
#include <string.h>
 main()
{
 /* Déclarations */
 char CH1[101], CH2[101]; /* chaînes données */
 char *P1, *P2; /* pointeurs d'aide dans CH1 et CH2 */
 int TROUVE; /* indicateur logique: vrai, si le caractère */
             /* actuel dans CH1 a été trouvé dans CH2.    */

 /* Saisie des données */
 printf("Entrez la première chaîne de caractères"
        " (max.100 caractères) :\n");
 gets(CH1);
 printf("Entrez la deuxième chaîne de caractères"
        " (max.100 caractères) :\n");
 gets(CH2);
 /* Eliminer les lettres communes */
 /* Idée: Parcourir CH2 de gauche à droite et contrôler  */
 /* pour chaque caractère s'il se trouve aussi dans CH1. */
 /* Si tel est le cas, éliminer le caractère de CH1 à    */
 /* l'aide de strcpy. */
 for (P2=CH2; *P2; P2++)
     {
      TROUVE = 0;
      for (P1=CH1 ; *P1 && !TROUVE ; P1++)
           if (*P2==*P1)
               {
                TROUVE = 1;
                strcpy(P1, P1+1);
               }
     }
  /* Affichage du résultat */
 printf("Chaîne résultat : \"%s\" \n", CH1);
 return 0;
}
```

Exercice 9.14

```
#include <stdio.h>
#include <string.h>
 main()
{
 /* Déclarations */
 char CH1[101], CH2[101]; /* chaînes données */
 char *P1, *P2; /* pointeurs d'aide dans CH1 et CH2 */
 int TROUVE; /* indicateur logique: vrai, si le caractère */
             /* actuel dans CH1 a été trouvé dans CH2.    */
```

```
 /* Saisie des données */
 printf("Entrez la chaîne à transformer"
        " (max.100 caractères) :\n");
 gets(CH1);
 printf("Entrez la chaîne à supprimer  "
        " (max.100 caractères) :\n");
 gets(CH2);
 /* Rechercher CH2 dans CH1 : */
 /* L'expression P2-CH2 est utilisée pour déterminer l'indice */
 /* de P2 dans CH2. On pourrait aussi résoudre le problème à  */
 /* l'aide d'un troisième pointeur P3 parcourant CH1. */
 TROUVE=0;
 for (P1=CH1 ; *P1 && !TROUVE ; P1++)
     {
      for (P2=CH2 ; *P2 == *(P1+(P2-CH2)) ; P2++)
          ;
      if (!*P2)
          TROUVE = 1;
     }

 /* A la fin de la boucle, P1 est incrémenté, donc */
 P1--;
 /* Si CH2 se trouve dans CH1, alors P1 indique la position */
 /* de la première occurence de CH2 dans CH1 et P2 pointe à */
 /* la fin de CH2. (P2-CH2) est alors la longueur de CH2.   */
 if (TROUVE)
     strcpy(P1, P1+(P2-CH2));

 /* Affichage du résultat */
 printf("Chaîne résultat : \"%s\" \n", CH1);
 return 0;
}
```

****

Les fonctions

Exercice 10.17 Tri de Shell

Traduire la fonction TRI_SHELL définie ci-dessous en C. Utiliser la fonction PERMUTER définie dans le cours.

Ecrire un programme profitant des fonctions définies dans les exercices précédents pour tester la fonction TRI_SHELL.

```
procédure TRI_SHELL(T,N)
|  (* Trie un tableau T d'ordre N par la méthode
|     de Shell en ordre croissant. *)
|  résultat: entier tableau T[100]
|  donnée: entier N
|  entier SAUT, M, K
|  booléen TERMINE
```

```
|  en SAUT ranger N
|  tant que (SAUT>1) faire
|  |  en SAUT ranger SAUT divent 2
|  |  répéter
|  |  |  en TERMINE ranger vrai
|  |  |  pour M variant de 1 à N-SAUT faire
|  |  |  |  en K ranger M+SAUT
|  |  |  |  si (T[M]>T[K]) alors
|  |  |  |  |   PERMUTER(T[M],T[K])
|  |  |  |  |   en TERMINE ranger faux
|  |  |  |  fsi
|  |  |  fpour
|  |  jusqu'à TERMINE
|  ftant (* SAUT <= 1 *)
fprocédure (* fin TRI_SHELL *)
```

Remarque: L'algorithme a été développé par D.L.Shell en 1959. En comparant d'abord des éléments très éloignés, l'algorithme a tendance à éliminer rapidement les grandes perturbations dans l'ordre des éléments. La distance entre les éléments qui sont comparés est peu à peu réduite jusqu'à 1. A la fin du tri, les éléments voisins sont arrangés.

Exercice 10.18

Déterminer le maximum de N éléments d'un tableau TAB d'entiers de trois façons différentes:

a) la fonction MAX1 retourne la valeur maximale

b) la fonction MAX2 retourne l'indice de l'élément maximal

c) la fonction MAX3 retourne l'adresse de l'élément maximal

Ecrire un programme pour tester les trois fonctions.

Exercice 10.19 Tri par sélection

Ecrire la fonction TRI_SELECTION qui trie un tableau de N entiers par la méthode de sélection directe du maximum (voir exercice 7.14). La fonction fera appel à la fonction PERMUTER (définie dans le cours) et à la fonction MAX3 (définie dans l'exercice précédent).

Ecrire un programme pour tester la fonction TRI_SELECTION.

Exercice 10.20

Ecrire la fonction INSERER qui place un élément X à l'intérieur d'un tableau qui contient N éléments triés par ordre croissant, de façon à obtenir un tableau à N+1 éléments triés par ordre croissant. La dimension du tableau est incrémentée dans la fonction INSERER.

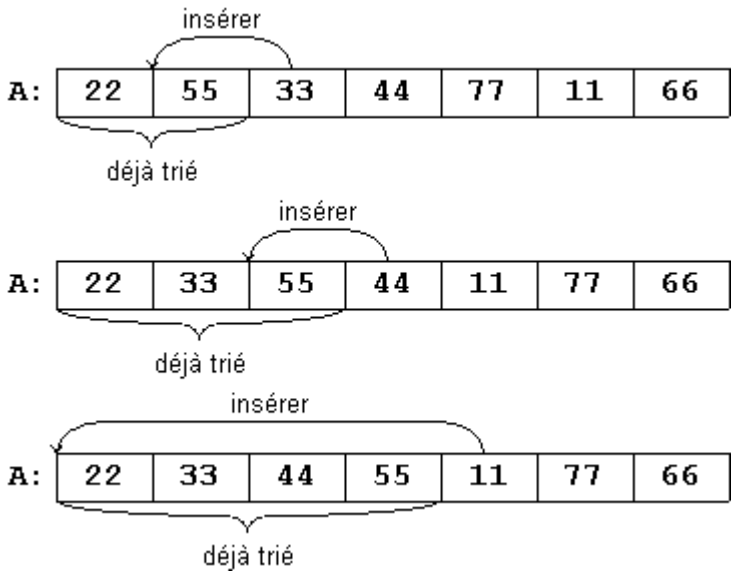
Ecrire un programme profitant des fonctions définies plus haut pour tester la fonction INSERER.

Exercice 10.21 Tri par insertion

Ecrire la fonction TRI_INSERTION qui utilise la fonction INSERER pour trier par ordre croissant les éléments d'un tableau à N éléments.

Ecrire un programme pour tester la fonction TRI_INSERTION.

Méthode: Trier le tableau de gauche à droite en insérant à chaque fois l'élément I+1 dans le tableau (déjà trié) des I premiers éléments.



Exercice 10.22

Ecrire la fonction RANGER qui arrange le contenu de ses deux paramètres X et Y de façon à ce que le contenu de X soit plus petit que celui de Y. RANGER retourne la valeur logique 1 si un échange a eu lieu, sinon 0.

Exercice 10.23 Tri par propagation

Ecrire la fonction TRI_BULLE qui trie un tableau de N éléments entiers par ordre croissant en appliquant la méthode de la bulle (tri par propagation - voir exercice 7.15). Employer la fonction RANGER de l'exercice ci-dessus.

Ecrire un programme pour tester la fonction TRI_BULLE.

Exercice 10.24 Fusion de tableaux triés

Ecrire la fonction FUSION qui construit un tableau FUS trié par ordre croissant avec les éléments de deux tableaux A et B triés par ordre croissant. Pour deux tableaux de dimensions N et M, le tableau FUS aura la dimension N+M. (Méthode: voir exercice 7.13)

Ecrire un programme qui teste la fonction FUSION à l'aide de deux tableaux lus au clavier et triés à l'aide de TRI_BULLE.

Exercice 10.17 Tri de Shell

```
#include <stdio.h>

main()
{
 /* Prototypes des fonctions appelées */
 void TRI_SHELL(int *T, int N);
 void LIRE_TAB (int *TAB, int *N, int NMAX);
 void ECRIRE_TAB (int *TAB, int N);
 /* Variables locales */
 int T[100]; /* Tableau d'entiers */
 int DIM;    /* Dimension du tableau */

 /* Traitements */
 LIRE_TAB (T, &DIM, 100);
 printf("Tableau donné : \n");
 ECRIRE_TAB (T, DIM);
 TRI_SHELL(T, DIM);
  printf("Tableau trié : \n");
 ECRIRE_TAB (T, DIM);
 return 0;
}

void TRI_SHELL(int *T, int N)
{
  /* Trie un tableau T d'ordre N par la méthode de Shell */
 /* Prototypes des fonctions appelées */
 void PERMUTER(int *A, int *B);
 /* Variables locales */
 int SAUT, M, K;
 int TERMINE;
 /* Traitements */
 SAUT = N;
 while (SAUT>1)
     {
      SAUT /= 2;
      do
         {
          TERMINE=1;
          for (M=0; M<N-SAUT; M++)  /* Attention aux indices! */
               {
                K=M+SAUT;
                if (*(T+M) > *(T+K))
                   {
                    PERMUTER(T+M,T+K);
```

```
                    TERMINE=0;
                  }
               }
            }
         while(!TERMINE); /* Attention: utiliser la négation de */
      }        /* la condition employée en lang algorithmique */
}

void PERMUTER(int *A, int *B)
{
 int AIDE;
 AIDE = *A;
 *A   = *B;
 *B   = AIDE;
}

void LIRE_TAB (int *TAB, int *N, int NMAX)
{
 . . .
}

void ECRIRE_TAB (int *TAB, int N)
{
 . . .
}
```

Exercice 10.18

Déterminer le maximum de N éléments d'un tableau TAB d'entiers de trois façons différentes:

a) la fonction MAX1 retourne la valeur maximale

```
int MAX1(int *TAB, int N)
{
 int MAX,I;  /* variables d'aide */
 MAX=*TAB;
 for (I=1; I<N; I++)
     if (MAX < *(TAB+I))
         MAX = *(TAB+I);
 return MAX;
}
```

b) la fonction MAX2 retourne l'indice de l'élément maximal

```
int MAX2(int *TAB, int N)
{
 int I,MAX; /* variables d'aide */
 MAX=0;
 for (I=1; I<N; I++)
         if (*(TAB+MAX) < *(TAB+I))
         MAX = I;
 return MAX;
}
```

c) la fonction MAX3 retourne l'adresse de l'élément maximal

```
int *MAX3(int *TAB, int N)
{
 int *MAX, *P; /* pointeurs d'aide */
 MAX=TAB;
 for (P=TAB; P<TAB+N; P++)
         if (*MAX < *P)
         MAX=P;
 return MAX;
}
```

Ecrire un programme pour tester les trois fonctions:

```
#include <stdio.h>

main()
{
 /* Prototypes des fonctions appelées */
 int MAX1 (int *TAB, int N);
 int MAX2 (int *TAB, int N);
 int *MAX3(int *TAB, int N);
 void LIRE_TAB (int *TAB, int *N, int NMAX);
 void ECRIRE_TAB (int *TAB, int N);
 /* Variables locales */
 int T[100]; /* Tableau d'entiers */
 int DIM;    /* Dimension du tableau */

 /* Traitements */
 LIRE_TAB (T, &DIM, 100);
 printf("Tableau donné : \n");
 ECRIRE_TAB (T, DIM);
 printf("MAX1 : %d \n",   MAX1(T,DIM)  );
 printf("MAX2 : %d \n", T[MAX2(T,DIM)] );
 printf("MAX3 : %d \n",  *MAX3(T,DIM)  );
 return 0;
}

int MAX1(int *TAB, int N)
{
 . . .
}

int MAX2(int *TAB, int N)
{
 . . .
}

int *MAX3(int *TAB, int N)
{
 . . .
}

void LIRE_TAB (int *TAB, int *N, int NMAX)
{
```

```
 . . .
}

void ECRIRE_TAB (int *TAB, int N)
{
 . . .
}
```

Exercice 10.19 Tri par sélection

```
#include <stdio.h>

main()
{
 /* Prototypes des fonctions appelées */
 void TRI_SELECTION(int *T, int N);
 void LIRE_TAB (int *TAB, int *N, int NMAX);
 void ECRIRE_TAB (int *TAB, int N);
 /* Variables locales */
 int T[100]; /* Tableau d'entiers */
 int DIM;    /* Dimension du tableau */


/* Traitements */
 LIRE_TAB (T, &DIM, 100);
 printf("Tableau donné : \n");
 ECRIRE_TAB (T, DIM);
 TRI_SELECTION(T, DIM);
  printf("Tableau trié : \n");
 ECRIRE_TAB (T, DIM);
 return 0;
}

void TRI_SELECTION(int *T, int N)
{
  /* Prototypes des fonctions appelées */
 void PERMUTER(int *A, int *B);
 int *MAX3(int *TAB, int N);
 /* Variables locales */
 int I;  /* rang à partir duquel T n'est pas trié */

 /* Tri par sélection directe du maximum */
 for (I=0 ; I<N-1 ; I++)
      PERMUTER(T+I, MAX3(T+I,N-I) );
}

int *MAX3(int *TAB, int N)
{
 . . .
}

void PERMUTER(int *A, int *B)
```

```
{
 . . .
}

void LIRE_TAB (int *TAB, int *N, int NMAX)
{
 . . .
}

void ECRIRE_TAB (int *TAB, int N)
{
 . . .
}
```

Exercice 10.20

```
#include <stdio.h>
main()
{
 /* Prototypes des fonctions appelées */
 void INSERER(int X, int *T, int *N);
 void LIRE_TAB (int *TAB, int *N, int NMAX);
 void ECRIRE_TAB (int *TAB, int N);
 /* Variables locales */
 int T[100]; /* Tableau d'entiers */
 int DIM;    /* Dimension du tableau */
 int A;      /* Nombre à insérer */
 /* Traitements */
 LIRE_TAB (T, &DIM, 100);
 printf("Tableau donné : \n");
 ECRIRE_TAB (T, DIM);
 printf("Introduire le nombre à insérer : ");
 scanf("%d", &A);
 INSERER(A, T, &DIM);
  printf("Tableau résultat : \n");
 ECRIRE_TAB (T, DIM);
 return 0;
}

void INSERER(int X, int *T, int *N)
{
  /* Variables locales */
 int I;
 /* Insertion de X dans le tableau T supposé trié: */
 /* Déplacer les éléments plus grands que X d'une  */
 /* position vers l'arrière. */
 for (I=*N ; I>0 && *(T+I-1)>X ; I--)
      *(T+I) = *(T+I-1);
  /* X est copié à la position du dernier élément déplacé */
 *(T+I)=X;
  /* Nouvelle dimension du tableau: */
 (*N)++; /* Attention aux parenthèses ! */
```

```
}

void LIRE_TAB (int *TAB, int *N, int NMAX)
{
 . . .
}

void ECRIRE_TAB (int *TAB, int N)
{
 . . .
}
```

Exercice 10.21 Tri par insertion

```
#include <stdio.h>

main()
{
 /* Prototypes des fonctions appelées */
 void TRI_INSERTION(int *T, int N);
 void LIRE_TAB (int *TAB, int *N, int NMAX);
 void ECRIRE_TAB (int *TAB, int N);
 /* Variables locales */
 int T[100]; /* Tableau d'entiers */
 int DIM;    /* Dimension du tableau */
 /* Traitements */
 LIRE_TAB (T, &DIM, 100);
 printf("Tableau donné : \n");
 ECRIRE_TAB (T, DIM);
 TRI_INSERTION(T, DIM);
  printf("Tableau trié : \n");
 ECRIRE_TAB (T, DIM);
 return 0;
}


void TRI_INSERTION(int *T, int N)
{
  void INSERER(int X, int *T, int *N);
 /* Variables locales */
 int I;  /* indice courant */
 /* Tri de T par insertion */
 I=1;
 while (I<N)
      INSERER(*(T+I), T, &I);
}

void INSERER(int X, int *T, int *N)
{
 . . .
}
```

```
void LIRE_TAB (int *TAB, int *N, int NMAX)
{
 . . .
}

void ECRIRE_TAB (int *TAB, int N)
{
 . . .
}
```

Exercice 10.22

```
int RANGER(int *X, int *Y)
{
 int AIDE;
 if (*X>*Y)
     {
      AIDE = *X;
      *X = *Y;
      *Y = AIDE;
      return 1;
      }
 else
     return 0;
 }
```

Exercice 10.23 Tri par propagation

```
#include <stdio.h>
main()
{
 /* Prototypes des fonctions appelées */
 void LIRE_TAB (int *TAB, int *N, int NMAX);
 void TRI_BULLE(int *T, int N);
 void ECRIRE_TAB (int *TAB, int N);
 /* Variables locales */
 int T[100]; /* Tableau d'entiers */
 int DIM;    /* Dimension du tableau */
 /* Traitements */
 LIRE_TAB (T, &DIM, 100);
 printf("Tableau donné : \n");
 ECRIRE_TAB (T, DIM);
 TRI_BULLE(T, DIM);
  printf("Tableau trié : \n");
 ECRIRE_TAB (T, DIM);
 return 0;
}

void TRI_BULLE(int *T, int N)
{
  /* Prototypes des fonctions appelées */
 int RANGER(int *X, int *Y);
 /* Variables locales */
```

```
 int I,J;  /* indices courants */
 int FIN;  /* position où la dernière permutation a eu lieu */
      /* permet de ne pas trier un sous-ensemble déjà trié. */
 /* Tri de T par propagation de l'élément maximal */
 for (I=N-1 ; I>0 ; I=FIN)
    {
     FIN=0;
     for (J=0; J<I; J++)
          if (RANGER(T+J, T+J+1)) FIN = J;
    }
}

int RANGER(int *X, int *Y)
{
 . . .
}
void LIRE_TAB (int *TAB, int *N, int NMAX)
{
 . . .
}
void ECRIRE_TAB (int *TAB, int N)
{
 . . .
}
```

Exercice 10.24 Fusion de tableaux triés

```
#include <stdio.h>

main()
{
 /* Prototypes des fonctions appelées */
 void FUSION(int *A, int *B, int *FUS, int N, int M);
 void TRI_BULLE(int *T, int N);
 void LIRE_TAB (int *TAB, int *N, int NMAX);
 void ECRIRE_TAB (int *TAB, int N);
 /* Variables locales */
 /* Les tableaux et leurs dimensions */
 int A[100], B[100], FUS[200];
 int N, M;
 /* Traitements */
 printf("*** Tableau A ***\n");
 LIRE_TAB (A, &N, 100);
 printf("*** Tableau B ***\n");
 LIRE_TAB (B, &M, 100);
 TRI_BULLE(A, N);
  printf("Tableau A trié : \n");
 ECRIRE_TAB (A, N);
 TRI_BULLE(B, M);
  printf("Tableau B trié : \n");
 ECRIRE_TAB (B, M);
 FUSION(A,B,FUS,N,M);
```

```
  printf("Tableau FUS : \n");
 ECRIRE_TAB (FUS, N+M);
 return 0;
}


void FUSION(int *A, int *B, int *FUS, int N, int M)
{
  /* Variables locales */
 /* Indices courants dans A, B et FUS */
 int IA,IB,IFUS;
 /* Fusion de A et B dans FUS */
 IA=0, IB=0; IFUS=0;
 while ((IA<N) && (IB<M))
    if (*(A+IA)<*(B+IB))
        {
         *(FUS+IFUS)=*(A+IA);
         IFUS++;
         IA++;
        }
    else
        {
         FUS[IFUS]=B[IB];
         IFUS++;
         IB++;
        }
  /* Si A ou B sont arrivés à la fin, alors */
 /* copier le reste de l'autre tableau.    */
 while (IA<N)
     {
      *(FUS+IFUS)=*(A+IA);
      IFUS++;
      IA++;
     }
 while (IB<M)
     {
      *(FUS+IFUS)=*(B+IB);
      IFUS++;
      IB++;
     }
}

void TRI_BULLE(int *T, int N)
{
 /* Prototypes des fonctions appelées */
 int RANGER(int *X, int *Y);
 . . .
}
int RANGER(int *X, int *Y)
{
 . . .
```

```
}
void LIRE_TAB (int *TAB, int *N, int NMAX)
{
 . . .
}
void ECRIRE_TAB (int *TAB, int N)
{
 . . .
}
```

Exercice 10.25

Ecrire la fonction LONG_CH qui retourne la longueur d'une chaîne de caractères CH comme résultat. Implémentez LONG_CH sans utiliser de variable d'aide numérique.

Exercice 10.26

Ecrire la fonction MAJ_CH qui convertit toutes les lettres d'une chaîne en majuscules, sans utiliser de variable d'aide.

Exercice 10.27

Ecrire la fonction AJOUTE_CH à deux paramètres CH1 et CH2 qui copie la chaîne de caractères CH2 à la fin de la chaîne CH1 sans utiliser de variable d'aide.

Exercice 10.28

Ecrire la fonction INVERSER_CH qui inverse l'ordre des caractères d'une chaîne CH. Utiliser la fonction LONG_CH (définie plus haut) et définir une fonction d'aide PERMUTER_CH qui échange les valeurs de deux variables du type **char**.

Exercice 10.29

Ecrire la fonction NMOTS_CH qui retourne comme résultat le nombre de mots contenus dans une chaîne de caractères CH. Utiliser une variable logique, la fonction **isspace** et une variable d'aide N.

Exercice 10.30

Ecrire la fonction MOT_CH qui retourne un pointeur sur le N-ième mot d'une chaîne CH s'il existe, sinon un pointeur sur le symbole de fin de chaîne.

Exercice 10.31

Ecrire la fonction EGAL_N_CH qui retourne la valeur 1 si les N premiers caractères de CH1 et CH2 sont égaux, sinon la valeur 0. (Si N est plus grand que la longueur de CH1 ou de CH2, le résultat peut être 1 ou 0).

Exercice 10.32

Utiliser la fonction EGAL_N_CH et LONG_CH pour écrire la fonction CHERCHE_CH qui retourne un pointeur sur la première apparition de CH1 dans CH2, sinon un pointeur sur le symbole de fin de chaîne.

Exercice 10.33

Ecrire la fonction CH_ENTIER qui retourne la valeur numérique d'une chaîne de caractères représentant un entier (positif ou négatif) du type **long**. Si la chaîne ne représente pas une valeur entière correcte, la fonction arrête la conversion et fournit la valeur qu'elle a su reconnaître jusqu'à ce point.

Exercice 10.34

Ecrire la fonction CH_DOUBLE qui retourne la valeur numérique d'une chaîne de caractères représentant un réel en *notation décimale*. Si la chaîne ne représente pas une valeur décimale correcte, la fonction arrête la conversion et fournit la valeur qu'elle a su reconnaître jusqu'à ce point. (Méthode: voir exercice 8.18.)

Exercice 10.35

Ecrire la fonction ENTIER_CH qui construit une chaîne de caractères CH qui représente un nombre entier N du type **long**. N et CH sont les paramètres de la fonction ENTIER_CH. Utiliser la fonction INVERSER_CH définie plus haut.

Exercice 10.25

```
int LONG_CH(char *CH)
{
 char *P;
```

```
 for (P=CH ; *P; P++) ;
 return P-CH;
}
```

Exercice 10.26

```
void MAJ_CH(char *CH)
{
 for ( ; *CH; CH++)
      if (*CH>='a' && *CH<='z')
          *CH = *CH-'a'+'A';
}
```

Exercice 10.27

```
void AJOUTE_CH(char *CH1, char *CH2)
{
 while (*CH1) /* chercher la fin de CH1 */
        CH1++;
 while (*CH2) /* copier CH2 à la fin de CH1 */
       {
        *CH1 = *CH2;
        CH1++;
        CH2++;
       }
 *CH1='\0'; /* terminer la chaîne CH1 */
}
```

Solution plus compacte :

```
void AJOUTE_CH(char *CH1, char *CH2)
{
 for ( ; *CH1 ; CH1++) ;
 for ( ; *CH1 = *CH2 ; CH1++, CH2++) ;
}
```

Comme la conditon d'arrêt est évaluée lors de l'affectation, le symbole de fin de chaîne est aussi copié.

Exercice 10.28

```
void INVERSER_CH (char *CH)
{
 /* Prototypes des fonctions appelées */
 int LONG_CH(char *CH);
 void PERMUTER_CH(char *A, char *B);
 /* Variables locales */
 int I,J;
 /* Inverser la chaîne par permutations successives */
 J = LONG_CH(CH)-1;
 for (I=0 ; I<J ; I++,J--)
      PERMUTER_CH(CH+I, CH+J);
}

void PERMUTER_CH(char *A, char *B)
{
 char AIDE;
```

```
 AIDE = *A;
 *A   = *B;
 *B   = AIDE;
}

int LONG_CH(char *CH)
{
 . . .
}
```

Exercice 10.29

```
#include <ctype.h>

int NMOTS_CH(char *CH)
{
 /* Variables locales */
 int N;         /* nombre de mots */
 int DANS_MOT; /* indicateur logique : */
               /* vrai si CH pointe à l'intérieur d'un mot */
 DANS_MOT=0;
 for (N=0; *CH; CH++)
     if (isspace(*CH))
         DANS_MOT=0;
     else if (!DANS_MOT)
         {
          DANS_MOT=1;
          N++;
         }
 return N;
}
```

Exercice 10.30

```
#include <ctype.h>

char *MOT_CH(int N, char *CH)
{
 /* Variables locales */
 int DANS_MOT; /* indicateur logique : */
               /* vrai si CH pointe à l'intérieur d'un mot */
 DANS_MOT=0;
 for ( ; N>0 && *CH ; CH++)
     if (isspace(*CH))
         DANS_MOT=0;
     else if (!DANS_MOT)
         {
          DANS_MOT=1;
          N--;
          CH--; /* Pour réajuster l'effet de l'incrémentation */
         }
 return CH;
}
```

Exercice 10.31

```
int EGAL_N_CH(int N, char *CH1, char *CH2)
{
 while (--N && *CH1==*CH2)
    {
     CH1++;
     CH2++;
    }
 return (*CH1==*CH2);
}
```

Exercice 10.32

```
char *CHERCHE_CH(char *CH1, char *CH2)
{
 /* Prototypes des fonctions appelées */
 int LONG_CH(char *CH);
 int EGAL_N_CH(int N, char *CH1, char *CH2);
 /* Variables locales */
 int L;
 /* Recherche de CH1 dans CH2 */
 L=LONG_CH(CH1);
 while (*CH2 && !EGAL_N_CH(L, CH1, CH2))
          CH2++;
 return CH2;
}

int LONG_CH(char *CH)
{
 . . .
}

int EGAL_N_CH(int N, char *CH1, char *CH2)
{
 . . .
}
```

Exercice 10.33

```
long CH_ENTIER(char *CH)
{
 /* Variables locales */
 long N;
 int SIGNE;
 /* Traitement du signe */
 SIGNE = 1;
 if (*CH=='-') SIGNE = -1;
 if (*CH=='-' || *CH=='+') CH++;
 /* Conversion des chiffres */
 for (N=0 ; *CH>='0' && *CH<='9' ; CH++)
         N = N*10 + (*CH-'0');
 return SIGNE*N;
}
```

Exercice 10.34

```
#include <ctype.h>
#include <math.h>

double CH_DOUBLE(char *CH)
{
 /* Variables locales */
 double N; /* résultat numérique */
 int SIGNE; /* signe de la valeur rationnelle */
 int DEC; /* positions derrière la virgule  */

 /* Initialisation des variables */
 N = 0.0;
 SIGNE = 1;
 /* Traitement du signe */
 if (*CH=='-') SIGNE = -1;
 if (*CH=='-' || *CH=='+') CH++;
 /* Positions devant le point décimal */
 for ( ; isdigit(*CH); CH++)
     N = N*10.0 + (*CH-'0');
 /* Traitement du point décimal */
 if (*CH=='.') CH++;

 /* Traitement des positions derrière le point décimal */
 for (DEC=0; isdigit(*CH); CH++)
    {
     N = N*10.0 + (*CH-'0');
     DEC++;
    }
 /* Calcul de la valeur à partir du signe et */
 /* du nombre de décimales. */
 return SIGNE*N/pow(10,DEC);
}
```

Exercice 10.35

```
#include <stdio.h>

void ENTIER_CH(long N, char *CH)
{
  /* Prototypes des fonctions appelées */
 void INVERSER_CH(char *CH);
 /* Variables locales */
 int I;
 int SIGNE;
 /* Traitement du signe */
 SIGNE = (N<0) ? -1 : 1;
 if (N<0) N=-N;
 /* Conversion des chiffres (à rebours) */
 I=0;
 do
```

```
    {
     *(CH+I) = N % 10 + '0';
     I++;
    }
 while ((N/=10) > 0);
 /* Ajouter le signe à la fin de la chaîne */
 if (SIGNE<0)
    {
     *(CH+I)='-';
     I++;
    }
 /* Terminer la chaîne */
 *(CH+I)='\0';
 /* Inverser la chaîne */
 INVERSER_CH(CH);
}

void INVERSER_CH (char *CH)
{
 /* Prototypes des fonctions appelées */
 int LONG_CH(char *CH);
 void PERMUTER_CH(char *A, char *B);
  . . .
}

int LONG_CH(char *CH)
{
  . . .
}

void PERMUTER_CH(char *A, char *B)
{
  . . .
}
```

****

Les fichiers

Exercice 11.1

Créer sur disquette puis afficher à l'écran le fichier INFORM.TXT dont les informations sont structurées de la manière suivante:

Numéro de matricule (entier)

Nom (chaîne de caractères)

Prénom (chaîne de caractères)

Le nombre d'enregistrements à créer est à entrer au clavier par l'utilisateur.

Exercice 11.2

Ecrire un programme qui crée sur disquette un fichier INFBIS.TXT qui est la copie exacte (enregistrement par enregistrement) du fichier INFORM.TXT.

Exercice 11.3

Ajouter un nouvel enregistrement (entré au clavier) à la fin de INFORM.TXT et sauver le nouveau fichier sous le nom INFBIS.TXT.

Exercice 11.4

Insérer un nouvel enregistrement dans INFORM.TXT en supposant que le fichier est trié relativement à la rubrique NOM et sauver le nouveau fichier sous le nom INFBIS.TXT.

Exercice 11.5

Supprimer dans INFORM.TXT tous les enregistrements:

a) dont le numéro de matricule se termine par 8

b) dont le prénom est "Paul" (utiliser **strcmp**)

c) dont le nom est un palindrome. Définir une fonction d'aide PALI qui fournit le résultat 1 si la chaîne transmise comme paramètre est un palindrome, sinon la valeur zéro.

Sauver le nouveau fichier à chaque fois sous le nom INFBIS.TXT.

Exercice 11.6

Créer sur disquette puis afficher à l'écran le fichier FAMILLE.TXT dont les informations sont structurées de la manière suivante:

Nom de famille

Prénom du père

Prénom de la mère

Nombre d'enfants

Prénoms des enfants

Le nombre d'enregistrements à créer est entré au clavier.

Attention: Le nombre de rubriques des enregistrements varie avec le nombre d'enfants !

Exercice 11.7

Ecrire un programme qui crée sur disquette le fichier MOTS.TXT contenant une série de 50 mots au maximum (longueur maximale d'un mot: 50 caractères). La saisie des mots se terminera à l'introduction du symbole `'*'` qui ne sera pas écrit dans le fichier.

Exercice 11.8

Ecrire un programme qui affiche le nombre de mots, le nombre de palindromes ainsi que la longueur moyenne des mots contenus dans le fichier MOTS.TXT. Utiliser les deux fonctions d'aide PALI et LONG_CH définies au chapitre 10.

Exercice 11.9

Ecrire un programme qui charge les mots du fichier MOTS.TXT dans la mémoire centrale, les trie d'après la méthode par propagation (méthode de la bulle - voir exercice 7.15) et les écrit dans un deuxième fichier MOTS_TRI.TXT sur la disquette. Les mots seront mémorisés à l'aide d'un tableau de pointeurs sur **char** et la mémoire nécessaire sera réservée de façon dynamique.

Exercice 11.10

A l'aide d'un éditeur de textes, créer un fichier NOMBRES.TXT qui contient une liste de nombres entiers. Dans le fichier, chaque nombre doit être suivi par un retour à la ligne. Ecrire un programme qui affiche les nombres du fichier, leur somme et leur moyenne.

Exercice 11.11

Ecrire un programme qui remplace, dans un fichier contenant un texte, les retours à la ligne par des espaces. Si plusieurs retours à la ligne se suivent, seulement le premier sera remplacé. Les noms des fichiers source et destination sont entrés au clavier.

Exercice 11.12

Ecrire un programme qui détermine dans un fichier un texte dont le nom est entré au clavier, le nombre de phrases terminées par un point, un point d'interrogation ou un point d'exclamation.

Utiliser une fonction d'aide FIN_PHRASE qui décide si un caractère transmis comme paramètre est un des séparateurs mentionnés ci-dessus. FIN_PHRASE retourne la valeur (logique) 1 si le caractère est égal à `'.'`, `'!'` ou `'?'` et 0 dans le cas contraire.

Exercice 11.13

Ecrire un programme qui détermine dans un fichier un texte dont le nom est entré au clavier:

- le nombre de caractères qu'il contient,

- le nombre de chacune des lettres de l'alphabet

(sans distinguer les majuscules et les minuscules),

- le nombre de mots,

- le nombre de paragraphes (c.-à-d.: des retours à la ligne),

Les retours à la ligne ne devront pas être comptabilisés dans les caractères. On admettra que deux mots sont toujours séparés par un ou plusieurs des caractères suivants:

- fin de ligne

- espace

- ponctuation: `. : , ; ? !`

- parenthèses: `( )`

- guillemets: `"`

- apostrophe: `'`

Utiliser une fonction d'aide SEPA qui décide si un caractère transmis comme paramètre est l'un des séparateurs mentionnés ci-dessus. SEPA restituera la valeur (logique) 1 si le caractère est un séparateur et 0 dans le cas contraire. SEPA utilise un tableau qui contient les séparateurs à détecter.

Exemple:

```
Nom du fichier texte : A:LITTERA.TXT
Votre fichier contient:
   12 paragraphes
   571 mots
   4186 caractères
dont
   279 fois la lettre a
   56 fois la lettre b
   . . .
   3 fois la lettre z
et 470 autres caractères
```

Exercice 11.14

Ecrire un programme qui affiche le contenu d'un fichier texte sur un écran de 25 lignes et 80 colonnes en attendant la confirmation de l'utilisateur (par 'Enter') après chaque page d'écran. Utiliser la fonction **getchar**.

Exercice 11.15

Ecrire un programme qui vérifie la validité d'une série de numéros de CCP mémorisés dans un fichier. Un numéro de CCP est composé de trois parties: un numéro de compte, un séparateur '-' et un numéro de contrôle. Un numéro de CCP est correct:

- si le reste de la division entière de la valeur devant le séparateur '-' par 97 est différent de zéro et égal à la valeur de contrôle.

- si le reste de la division par 97 est zéro et la valeur de contrôle est 97.

Exemple:

Nombre de CCP 15742-28 :	15742 modulo 97 = 28	correct
Nombre de CCP 72270-5 :	72270 modulo 97 = 5	correct
Nombre de CCP 22610-10 :	22610 modulo 97 = 9	incorrect
Nombre de CCP 50537-0 :	50537 modulo 97 = 0	
	nombre incorrect, car la valeur de contrôle devrait être 97.	
Nombre de CCP 50537-97 :	50537 modulo 97 = 0	correct

Utiliser une fonction CCP_TEST qui obtient comme paramètres les deux parties numériques d'un nombre de CCP et qui affiche alors un message indiquant si le numéro de CCP est valide ou non.

Pour tester le programme, créer à l'aide d'un éditeur de texte un fichier CCP.TXT qui contient les numéros ci-dessus, suivis par des retours à la ligne.

Exercice 11.16

Deux fichiers FA et FB dont les noms sont à entrer au clavier contiennent des nombres entiers triés dans l'ordre croissant. Ecrire un programme qui copie le contenu de FA et FB respectivement dans les tableaux TABA et TABB dans la mémoire centrale. Les tableaux TABA et TABB sont fusionnés dans un troisième tableau trié en ordre croissant TABC. Après la fusion, la tableau TABC est sauvé dans un fichier FC dont le nom est à entrer au clavier.

La mémoire pour TABA, TABB et TFUS dont les nombres d'éléments sont inconnus, est réservée dynamiquement après que les longueurs des fichiers FA et FB ont été détectées.

Exercice 11.1

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

```
#include <stdlib.h>
main()
{
  /* Déclarations : */
  /* Nom du fichier et pointeur de référence */
  char NOM_FICH[] = "A:\\INFORM.TXT";
  FILE *FICHIER;
   /* Autres variables */
  char NOM[30], PRENOM[30];
  int MATRICULE;
  int I,N_ENR;

  /* Ouverture du nouveau fichier en écriture */
  FICHIER = fopen(NOM_FICH, "w");
  if (!FICHIER)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOM_FICH);
      exit(-1);
     }
   /* Saisie des données et création du fichier */
  printf("*** Création du fichier %s ***\n", NOM_FICH);
  printf("Nombre d'enregistrements à créer : ");
  scanf("%d",&N_ENR);
  for (I=1; I<=N_ENR; I++)
    {
     printf("Enregistrement No: %d \n", I);
     printf("Numéro de matricule : ");
     scanf("%d",&MATRICULE);
     printf("Nom    : ");
     scanf("%s",NOM);
     printf("Prénom : ");
     scanf("%s",PRENOM);
     fprintf(FICHIER, "%d\n%s\n%s\n", MATRICULE, NOM, PRENOM);
     }
  /* Fermeture du fichier */
  fclose(FICHIER);

  /* Ouverture du fichier en lecture */
  FICHIER = fopen(NOM_FICH, "r");
  if (!FICHIER)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOM_FICH);
      exit(-1);
     }
   /* Affichage du fichier */
  printf("*** Contenu du fichier  %s ***\n", NOM_FICH);
  while (!feof(FICHIER))
    {
     fscanf(FICHIER, "%d\n%s\n%s\n", &MATRICULE, NOM, PRENOM);
```

```
      printf("Matricule : %d\t", MATRICULE);
      printf("Nom et prénom : %s %s\n", NOM, PRENOM);
     }
  /* Fermeture du fichier */
  fclose(FICHIER);
   return 0;
}
```

Exercice 11.2

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char ANCIEN[]  = "A:\\INFORM.TXT";
  char NOUVEAU[] = "A:\\INFBIS.TXT";
  FILE *INFILE, *OUTFILE;
   /* Autres variables */
  char NOM[30], PRENOM[30];
  int MATRICULE;

  /* Ouverture de l'ancien fichier en lecture */
  INFILE = fopen(ANCIEN, "r");
   if (!INFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", ANCIEN);
      exit(-1);
     }
  /* Ouverture du nouveau fichier en écriture */
  OUTFILE = fopen(NOUVEAU, "w");
   if (!OUTFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOUVEAU);
      exit(-1);
     }

  /* Copie de tous les enregistrements */
  while (!feof(INFILE))
    {
     fscanf (INFILE, "%d\n%s\n%s\n", &MATRICULE, NOM, PRENOM);
     fprintf(OUTFILE, "%d\n%s\n%s\n", MATRICULE, NOM, PRENOM);
    }
 /* Fermeture des fichiers */
 fclose(OUTFILE);
 fclose(INFILE);
  return 0;
}
```

Exercice 11.3

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char ANCIEN[]  = "A:\\INFORM.TXT";
  char NOUVEAU[] = "A:\\INFBIS.TXT";
  FILE *INFILE, *OUTFILE;
  /* Autres variables */
  char NOM[30], PRENOM[30];

  int MATRICULE;
  char NOM_NOUV[30], PRE_NOUV[30];
  int MATRI_NOUV;
  /* Ouverture de l'ancien fichier en lecture */
  INFILE = fopen(ANCIEN, "r");
   if (!INFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", ANCIEN);
      exit(-1);
     }
  /* Ouverture du nouveau fichier en écriture */
  OUTFILE = fopen(NOUVEAU, "w");
   if (!OUTFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOUVEAU);
      exit(-1);
     }

  /* Saisie de l'enregistrement à ajouter */
  printf("Enregistrement à ajouter : \n");
  printf("Numéro de matricule : ");
  scanf("%d",&MATRI_NOUV);
  printf("Nom    : ");
  scanf("%s",NOM_NOUV);
  printf("Prénom : ");
  scanf("%s",PRE_NOUV);
  /* Copie des enregistrements de l'ancien fichier */
  while (!feof(INFILE))
    {
     fscanf (INFILE, "%d\n%s\n%s\n", &MATRICULE, NOM, PRENOM);
     fprintf(OUTFILE, "%d\n%s\n%s\n", MATRICULE, NOM, PRENOM);
    }
  /* Ecriture du nouvel enregistrement à la fin du fichier */
  fprintf(OUTFILE,"%d\n%s\n%s\n",MATRI_NOUV,NOM_NOUV,PRE_NOUV);
  /* Fermeture des fichiers */
```

```
  fclose(OUTFILE);
  fclose(INFILE);
   return 0;
}
```

Exercice 11.4

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
main()
{
  /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char ANCIEN[]  = "A:\\INFORM.TXT";
  char NOUVEAU[] = "A:\\INFBIS.TXT";
  FILE *INFILE, *OUTFILE;
  /* Autres variables */
  char NOM[30], PRENOM[30];
  int MATRICULE;
  char NOM_NOUV[30], PRE_NOUV[30];
  int MATRI_NOUV;
  int TROUVE;

  /* Ouverture de l'ancien fichier en lecture */
  INFILE = fopen(ANCIEN, "r");
   if (!INFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", ANCIEN);
      exit(-1);
     }

  /* Ouverture du nouveau fichier en écriture */
  OUTFILE = fopen(NOUVEAU, "w");
   if (!OUTFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOUVEAU);
      exit(-1);
     }

  /* Saisie de l'enregistrement à insérer */
  printf("Enregistrement à ajouter : \n");
  printf("Numéro de matricule : ");
  scanf("%d",&MATRI_NOUV);
  printf("Nom    : ");
  scanf("%s",NOM_NOUV);
  printf("Prénom : ");
  scanf("%s",PRE_NOUV);

  /* Copie des enregistrements dont le nom */
```

```
  /* précéde lexicogr. celui à insérer.    */
  TROUVE = 0;
  while (!feof(INFILE) && !TROUVE)
    {
     fscanf (INFILE, "%d\n%s\n%s\n", &MATRICULE, NOM, PRENOM);
     if (strcmp(NOM, NOM_NOUV) > 0)
         TROUVE = 1;
     else
         fprintf(OUTFILE, "%d\n%s\n%s\n", MATRICULE,NOM,PRENOM);
    }

  /* Ecriture du nouvel enregistrement, */
  fprintf(OUTFILE,"%d\n%s\n%s\n",MATRI_NOUV,NOM_NOUV,PRE_NOUV);
  /* et du dernier enregistrement lu. */
  if (TROUVE)
      fprintf(OUTFILE, "%d\n%s\n%s\n", MATRICULE, NOM, PRENOM);

  /* Copie du reste des enregistrements */
  while (!feof(INFILE))
    {
     fscanf (INFILE, "%d\n%s\n%s\n", &MATRICULE, NOM, PRENOM);
     fprintf(OUTFILE, "%d\n%s\n%s\n", MATRICULE, NOM, PRENOM);
    }

  /* Fermeture des fichiers */
  fclose(OUTFILE);
  fclose(INFILE);
   return 0;
}
```

Exercice 11.5

a) Supprimer les enregistrements, dont le numéro de matricule se termine par 8

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char ANCIEN[]  = "A:\\INFORM.TXT";
  char NOUVEAU[] = "A:\\INFBIS.TXT";
  FILE *INFILE, *OUTFILE;
  /* Autres variables */
  char NOM[30], PRENOM[30];
  int MATRICULE;

  /* Ouverture de l'ancien fichier en lecture */
  INFILE = fopen(ANCIEN, "r");
   if (!INFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", ANCIEN);
```

```
      exit(-1);
     }
  /* Ouverture du nouveau fichier en écriture */
  OUTFILE = fopen(NOUVEAU, "w");
   if (!OUTFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOUVEAU);
      exit(-1);
     }
  /* Copie de tous les enregistrements à l'exception */
  /* de ceux dont le numéro de matricule se termine  */
  /* par 8. */
  while (!feof(INFILE))
    {
     fscanf (INFILE, "%d\n%s\n%s\n", &MATRICULE, NOM, PRENOM);
     if (MATRICULE%10 != 8)
        fprintf(OUTFILE, "%d\n%s\n%s\n", MATRICULE,NOM,PRENOM);
    }
  /* Fermeture des fichiers */
  fclose(OUTFILE);
  fclose(INFILE);
   return 0;
}
```

b) Supprimer les enregistrements, dont le prénom est "Paul" (utiliser **strcmp**)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
 main()
{
  /* Déclarations */
  . . .
  /* Ouverture de l'ancien fichier en lecture */
  . . .
  /* Ouverture du nouveau fichier en écriture */
  . . .
  /* Copie de tous les enregistrements à l'exception */
  /* de ceux dont le prénom est 'Paul'. */
  while (!feof(INFILE))
    {
     fscanf (INFILE, "%d\n%s\n%s\n", &MATRICULE, NOM, PRENOM);
     if (strcmp(PRENOM,"Paul") != 0)
        fprintf(OUTFILE, "%d\n%s\n%s\n", MATRICULE,NOM,PRENOM);
    }
   /* Fermeture des fichiers */
  . . .
}
```

c) Supprimer les enregistrements, dont le nom est un palindrome. Définir une fonction d'aide PALI qui fournit le résultat 1 si la chaîne transmise comme paramètre est un palindrome, sinon la valeur zéro.

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

```
#include <stdlib.h>
main()
{
  /* Prototype de la fonction PALI */
  int PALI(char *CH);
   /* Déclarations */
  . . .
  /* Ouverture de l'ancien fichier en lecture */
   . . .
  /* Ouverture du nouveau fichier en écriture */
  . . .
  /* Copie de tous les enregistrements à l'exception */
  /* des palindromes. */
  while (!feof(INFILE))
    {
     fscanf (INFILE, "%d\n%s\n%s\n", &MATRICULE, NOM, PRENOM);
     if (!PALI(NOM))
        fprintf(OUTFILE, "%d\n%s\n%s\n", MATRICULE,NOM,PRENOM);
    }
   /* Fermeture des fichiers */
  . . .
}


int PALI(char *CH)
{
 /* Variable locale */
 char *CH2;
 /* Placer CH2 à la fin de la chaîne */
 for (CH2=CH; *CH2; CH2++)
     ;
 CH2--;
 /* Contrôler si la chaîne désignée par CH est un palindrome */
 for (; CH<CH2; CH++,CH2--)
      if (*CH != *CH2) return 0;
 return 1;
}
```

Exercice 11.6

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  /* Déclarations : */
  /* Nom du fichier et pointeur de référence */
  char NOM_FICH[] = "A:\\FAMILLE.TXT";
  FILE *FICHIER;
  /* Autres variables */
  char NOM[30], PERE[30], MERE[30], ENFANT[30];
  int J,N_ENFANTS;
  int I,N_ENR;
```

```
/* Ouverture du nouveau fichier en écriture */
FICHIER = fopen(NOM_FICH, "w");
 if (!FICHIER)
   {
    printf("\aERREUR: Impossible d'ouvrir "
           "le fichier: %s.\n", NOM_FICH);
    exit(-1);
   }

/* Saisie des données et création du fichier */
printf("*** Création du fichier %s ***\n", NOM_FICH);
printf("Nombre d'enregistrements à créer : ");
scanf("%d",&N_ENR);
for (I=1; I<=N_ENR; I++)
  {
   printf("Enregistrement No: %d \n", I);
   printf("Nom de famille : ");
   scanf("%s", NOM);
   printf("Prénom du père : ");
   scanf("%s", PERE);
   printf("Prénom de la mère : ");
   scanf("%s", MERE);
   printf("Nombre d'enfants : ");
   scanf("%d", &N_ENFANTS);
   fprintf(FICHIER, "%s\n%s\n%s\n%d\n",
                                NOM, PERE, MERE, N_ENFANTS);
   for (J=1; J<=N_ENFANTS; J++)
       {
        printf("Prénom %d. enfant : ", J);
        scanf("%s", ENFANT);
        fprintf(FICHIER, "%s\n", ENFANT);
       }
  }
/* Fermeture du fichier */
fclose(FICHIER);
/* Réouverture du fichier */
FICHIER = fopen(NOM_FICH, "r");
 if (!FICHIER)
   {
    printf("\aERREUR: Impossible d'ouvrir "
           "le fichier: %s.\n", NOM_FICH);
    exit(-1);
   }
/* Affichage du fichier */
printf("*** Contenu du fichier  %s ***\n", NOM_FICH);
while (!feof(FICHIER))
  {
   fscanf (FICHIER, "%s\n%s\n%s\n%d\n",
                                NOM, PERE, MERE, &N_ENFANTS);
    printf("\n");
```

```
      printf("Nom de famille : %s\n", NOM);
      printf("Nom du père    : %s %s\n", PERE, NOM);
      printf("Nom de la mère : %s %s\n", MERE, NOM);
      printf("Noms des enfants : \n", N_ENFANTS);
      for (J=1; J<=N_ENFANTS; J++)
          {
           fscanf(FICHIER, "%s\n", ENFANT);
           printf("\t%d. : %s %s\n", J, ENFANT, NOM);
          }
    }
  /* Fermeture du fichier */
  fclose(FICHIER);
   return 0;
}
```

Exercice 11.7

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  /* Déclarations : */
  /* Nom du fichier et pointeur de référence */
  char NOM_FICH[] = "A:\\MOTS.TXT";
  FILE *FICHIER;
  /* Autres variables */
  char CHAINE[50];

  /* Ouverture du nouveau fichier en écriture */
  FICHIER = fopen(NOM_FICH, "w");
   if (!FICHIER)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOM_FICH);
      exit(-1);
     }
  /* Saisie des données et création du fichier */
  printf("*** Création du fichier %s ***\n", NOM_FICH);
  do
    {
     printf("Entrez un mot ('*' pour finir) : ");
     scanf("%s", CHAINE);
     if (CHAINE[0] != '*')
         fprintf(FICHIER, "%s\n", CHAINE);
    }
  while (CHAINE[0] != '*');
  /* Fermeture du fichier */
  fclose(FICHIER);
   return 0;
}
```

Exercice 11.8

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  /* Prototypes des fonctions appelées */
  int PALI(char *CH);
  int LONG_CH(char *CH);
   /* Déclarations : */
  /* Nom du fichier et pointeur de référence */
  char NOM_FICH[] = "A:\\MOTS.TXT";
  FILE *FICHIER;
  /* Autres variables */
  char CHAINE[50];
  int N_PALI; /* nombre de palindromes */
  int N_MOTS; /* nombre de mots */
  int L_TOT;  /* longueur totale de tous les mots */

  /* Ouverture du nouveau fichier en écriture */
  FICHIER = fopen(NOM_FICH, "r");
   if (!FICHIER)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOM_FICH);
      exit(-1);
     }
  /* Compter les palindromes et accumuler */
  /* les longueurs des mots. */
  L_TOT =0;
  N_PALI=0;
  N_MOTS=0;
  while (!feof(FICHIER))
    {
     fscanf(FICHIER, "%s\n", CHAINE);
     N_MOTS++;
     L_TOT  += LONG_CH(CHAINE);
     N_PALI += PALI(CHAINE);
    }
   /* Fermeture du fichier */
  fclose(FICHIER);
   /* Affichage des résultats */
  printf("Le fichier %s contient :\n", NOM_FICH);
  printf("\t%d \tmots d'une longueur moyenne de :\n", N_MOTS);
  printf("\t%.1f \tcaractères et\n", (float)L_TOT/N_MOTS);
  printf("\t%d \tpalindromes\n", N_PALI);
  return 0;
}


int PALI(char *CH)
```

```
{
 /* Variable locale */
 char *CH2;
 /* Placer CH2 à la fin de la chaîne */
 for (CH2=CH; *CH2; CH2++)
     ;
 CH2--;
 /* Contrôler si la chaîne désignée par CH est un palindrome */
 for (; CH<CH2; CH++,CH2--)
      if (*CH != *CH2) return 0;
 return 1;
}


int LONG_CH(char *CH)
{
 char *P;
 for (P=CH ; *P; P++) ;
 return P-CH;
}
```

Exercice 11.9

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
main()
{
  /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char ANCIEN[]  = "A:\\MOTS.TXT";
  char NOUVEAU[] = "A:\\MOTS_TRI.TXT";
  FILE *INFILE, *OUTFILE;
  /* Tableau de pointeurs */
  char *TAB[50];
  /* Autres variables */
  char CHAINE[50];
  char *AIDE; /* pour la permutation */
  int N_MOTS; /* nombre de mots du fichier */
  int I; /* ligne à partir de laquelle TAB est trié */
  int J; /* indice courant */
  int FIN; /* ligne où la dernière permutation a eu lieu */
    /* permet de ne pas trier un sous-ensemble déjà trié */


  /* Ouverture de l'ancien fichier en lecture */
  INFILE = fopen(ANCIEN, "r");
   if (!INFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", ANCIEN);
      exit(-1);
```

```
    }
  /* Initialisation du du compteur des mots */
  N_MOTS = 0;
  /* Lecture du fichier dans la mémoire centrale */
  while (!feof(INFILE))
     {
      fscanf (INFILE, "%s\n", CHAINE);
      /* Réservation de la mémoire */
      TAB[N_MOTS] = malloc(strlen(CHAINE)+1);
      if (TAB[N_MOTS])
          strcpy(TAB[N_MOTS], CHAINE);
      else
          {
           printf("\aPas assez de mémoire \n");
           exit(-1);
          }
      N_MOTS++;
     }
  /* Fermeture du fichier */
  fclose(INFILE);
  /* Tri du tableau par propagation de l'élément maximal. */
  for (I=N_MOTS-1 ; I>0 ; I=FIN)
      {
       FIN=0;
       for (J=0; J<I; J++)
           if (strcmp(TAB[J],TAB[J+1])>0)
             {
              FIN=J;
              AIDE     = TAB[J];
              TAB[J]   = TAB[J+1];
              TAB[J+1] = AIDE;
             }
      }
   /* Ouverture du nouveau fichier en écriture */
  OUTFILE = fopen(NOUVEAU, "w");
   if (!OUTFILE)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOUVEAU);
      exit(-1);
     }
  /* Copie du tableau dans le nouveau fichier */
  for (I=0; I<N_MOTS; I++)
       fprintf(OUTFILE, "%s\n", TAB[I]);
  /* Fermeture du fichier */
  fclose(OUTFILE);
   return 0;
}
```

Exercice 11.10

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

```
#include <stdlib.h>
main()
{
  /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char NOM_FICH[]  = "A:\\NOMBRES.TXT";
  FILE *FICHIER;
  /* Autres variables */
  int NOMBRE; /* nombre actuel lu dans le fichier */
  int N;      /* compteur des nombres */
  long SOMME; /* somme des nombres */

  /* Ouverture de l'ancien fichier en lecture */
  FICHIER = fopen(NOM_FICH, "r");
   if (!FICHIER)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOM_FICH);
      exit(-1);
     }
  /* Lecture du fichier et comptabilité */
  N=0;
  SOMME=0;
  while (!feof(FICHIER))
     {
      fscanf (FICHIER, "%d\n", &NOMBRE);
      SOMME += NOMBRE;
      N++;
     }
  /* Fermeture du fichier */
  fclose(FICHIER);
   /* Affichage des résultats */
  printf("Le fichier %s contient %d nombres.\n", NOM_FICH, N);
  printf("La somme des nombres est   : %ld\n", SOMME);
  printf("La moyenne des nombres est : %f\n", (float)SOMME/N);
  return 0;
}
```

Exercice 11.11

```
#include <stdio.h>

main()
{
  /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char ANCIEN[30], NOUVEAU[30];
  FILE *INFILE, *OUTFILE;
  /* Autres variables */
  char C; /* caractère lu dans le fichier */
  char N_RET; /* Compteur des retours à la ligne consécutifs */
```

```
/* Ouverture de l'ancien fichier en lecture */
do
  {
   printf("Nom du fichier source : ");
   scanf("%s", ANCIEN);
   INFILE = fopen(ANCIEN, "r");
   if (!INFILE)
       printf("\aERREUR: Impossible d'ouvrir "
              "le fichier: %s.\n", ANCIEN);
  }
while (!INFILE);
 /* Ouverture du nouveau fichier en écriture */
do
  {
   printf("Nom du nouveau fichier : ");
   scanf("%s", NOUVEAU);
   OUTFILE = fopen(NOUVEAU, "w");
   if (!OUTFILE)
       printf("\aERREUR: Impossible d'ouvrir "
              "le fichier: %s.\n", NOUVEAU);
  }
while (!OUTFILE);


/* Copier tous les caractères et remplacer le */
/* premier retour à la ligne d'une suite par  */
/* un espace. */
N_RET=0;
while (!feof(INFILE))
  {
   C=fgetc(INFILE);
   if (!feof(INFILE))
      {
       if (C == '\n')
          {
           N_RET++;
           if (N_RET > 1)
              fputc('\n', OUTFILE);
           else
              fputc(' ', OUTFILE);
          }
       else
          {
           N_RET=0;
           fputc(C, OUTFILE);
          }
      }
  }
 /* Fermeture des fichiers */
fclose(OUTFILE);
fclose(INFILE);
```

```
   return 0;
}
```

Remarque :

Après la lecture par **fgetc**, il faut s'assurer encore une fois que le caractère lu est différent de **EOF**. Nous obtenons ainsi une construction un peu lourde:

```
   while (!feof(INFILE))
     {
      C=fgetc(INFILE);
      if (!feof(INFILE))
         {
          . . .
         }
     }
```

Il est possible de réunir plusieurs instructions dans le bloc conditionnel de la structure **while**, en les séparant par des virgules. En pratique, on retrouve cette solution souvent pour éviter des constructions inutilement lourdes:

```
   while (C=fgetc(INFILE), !feof(INFILE))
     {
          . . .
     }
```

Exercice 11.12

```
#include <stdio.h>
main()
{
  /* Prototype de la fonction FIN_PHRASE */
  int FIN_PHRASE(char C);
   /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char NOM_FICH[30];
  FILE *FICHIER;
  /* Autres variables */
  char C; /* caractère lu dans le fichier */
  char NP; /* Compteur de phrases */

  /* Ouverture de l'ancien fichier en lecture */
  do
    {
     printf("Nom du fichier texte : ");
     scanf("%s", NOM_FICH);
     FICHIER = fopen(NOM_FICH, "r");
      if (!FICHIER)
         printf("\aERREUR: Impossible d'ouvrir "
                "le fichier: %s.\n", NOM_FICH);
    }
  while (!FICHIER);
  /* Compter les symboles de fin de phrase */
  NP=0;
  while (!feof(FICHIER))
        NP += FIN_PHRASE(fgetc(FICHIER));
```

```
   /* Fermeture du fichier */
  fclose(FICHIER);
   /* Affichage du résultat */
  printf("Le fichier %s contient %d phrases.\n",
                                                  NOM_FICH, NP);
  return 0;
}

int FIN_PHRASE(char C)
{
 return (C=='.' || C=='!' || C=='?');
}
```

Exercice 11.13

```
#include <stdio.h>
main()
{
  /* Prototype de la fonction FIN_PHRASE */
  int SEPA(char C);
   /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char NOM_FICH[30];
  FILE *FICHIER;

  /* Autres variables */
  char C;       /* caractère lu dans le fichier */
  int ABC[26];  /* compteurs des lettres de l'alphabet */
   int NTOT;     /* nombre total des caractères */
  int NAUTRES;  /* nombre des caractères qui ne font pas
                   partie de l'alphabet */
  int NMOTS;    /* nombre des mots */
  int NPARA;    /* nombre de paragraphes (retours à la ligne) */
  int I;        /* indice d'aide */
  int DANS_MOT; /* indicateur logique: */
                /* vrai si la tête de lecture se trouve */
                /* actuellement à l'intérieur d'un mot. */

  /* Ouverture de l'ancien fichier en lecture */
  do
    {
     printf("Nom du fichier texte : ");
     scanf("%s", NOM_FICH);
     FICHIER = fopen(NOM_FICH, "r");
      if (!FICHIER)
         printf("\aERREUR: Impossible d'ouvrir "
                "le fichier: %s.\n", NOM_FICH);
    }
  while (!FICHIER);
  /* Initialisations des variables */
  for (I=0; I<26; I++)
      ABC[I]=0;
```

```
 NTOT    =0;
NAUTRES =0;
NMOTS   =0;
NPARA   =0;
DANS_MOT=0;
/* Examination des caractères du fichier */
while (!feof(FICHIER))
   {
    C=fgetc(FICHIER);
    if (!feof(FICHIER))
       {
        /* Comptage au niveau caractères */
        if (C=='\n')
           NPARA++;
        else
           {
            NTOT++;
            if (C>='a' && C<='z')
               ABC[C-'a']++;
            else if (C>='A' && C<='Z')
               ABC[C-'A']++;
            else
               NAUTRES++;
           }


        /* Comptage des mots */
        if (SEPA(C))
           {
            if (DANS_MOT)
               {
                NMOTS++;
                DANS_MOT=0;
               }
           }
        else
           DANS_MOT=1;
       }
   }
 /* Fermeture du fichier */
fclose(FICHIER);
 /* Affichage du résultat */
printf("Votre fichier contient :\n");
printf("\t%d paragraphes\n", NPARA);
printf("\t%d mots\n", NMOTS);
printf("\t%d caractères\ndont\n", NTOT);
for (I=0; I<26; I++)
     printf("\t%d fois la lettre %c\n", ABC[I], 'a'+I);
printf("et %d autres caractères\n", NAUTRES);
return 0;
}
```

```
int SEPA(char C)
{
 /* Tableau contenant tous les séparateurs de mots */
 char SEP[12] = { '\n', ' ', ',', ';', '.', ':',
                  '?', '!', '(', ')', '"', '\'' };
 int I;

 /* Comparaison de C avec tous les éléments du tableau */
 for (I=0 ; C!=SEP[I] && I<12 ; I++) ;
 if (I==12)
     return 0;
 else
     return 1;
  /* ou bien simplement : */
 /* return (I != 12);    */
}
```

Exercice 11.14

```
#include <stdio.h>
main()
{
  /* Noms des fichiers et pointeurs de référence */
  char NOM_FICH[30];
  FILE *FICHIER;
  /* Autres variables */
  char C;  /* caractère actuel */
  int NLIGNE, NCOLO; /* position actuelle sur l'écran */

  /* Ouverture de l'ancien fichier en lecture */
  do
    {
     printf("Nom du fichier texte : ");
     scanf("%s", NOM_FICH);
     FICHIER = fopen(NOM_FICH, "r");
      if (!FICHIER)
         printf("\aERREUR: Impossible d'ouvrir "
                "le fichier: %s.\n", NOM_FICH);
    }
  while (!FICHIER)  ;
  getchar();
  /* Initialisations des variables */
  NLIGNE  = 0; NCOLO   = 0;
  /* Affichage du fichier */
  while (!feof(FICHIER))
     {
      C=fgetc(FICHIER);
      if (!feof(FICHIER))
         {
          NCOLO++;
```

```
          if (NCOLO==80 || C=='\n') /* fin de la ligne */
             {
              NLIGNE++;
              if (NLIGNE==25) /* fin de l'écran */
                 {
                  getchar();
                  NLIGNE=1;
                 }
              printf("%c",C);
              NCOLO=1;
             }
           else
              printf("%c",C);
         }
     }
   /* Fermeture du fichier */
  fclose(FICHIER);
   return 0;
}
```

Exercice 11.15

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  /* Prototype de la fonction CCP_TEST */
  void CCP_TEST(long COMPTE, int CONTROLE);
   /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char NOM_FICH[]  = "A:\\CCP.TXT";
  FILE *FICHIER;
  /* Autres variables */
  long COMPTE;  /* nombre du compte CCP */
  int CONTROLE; /* nombre de contrôle   */

  /* Ouverture du fichier CCP.TXT en lecture */
  FICHIER = fopen(NOM_FICH, "r");
   if (!FICHIER)
     {
      printf("\aERREUR: Impossible d'ouvrir "
             "le fichier: %s.\n", NOM_FICH);
      exit(-1);
     }


  /* Lecture des nombres et appel de la fonction CCP_TEST  */
  /* A l'aide de la chaîne de format, scanf lit les deux   */
  /* parties du nombre de CCP, les convertit en long resp. */
  /* en int et affecte les résultats aux variables COMPTE  */
  /* et CONTROLE. */
  while (!feof(FICHIER))
```

```
      {
       fscanf (FICHIER, "%ld-%d\n", &COMPTE, &CONTROLE);
       CCP_TEST(COMPTE, CONTROLE);
      }
   /* Fermeture du fichier */
  fclose(FICHIER);
   return 0;
}


void CCP_TEST(long COMPTE, int CONTROLE)
{
 int RESTE;
 RESTE = COMPTE % 97;
 if (RESTE == 0)
     RESTE = 97;
 if (RESTE == CONTROLE)
     printf ("Le nombre CCP %ld-%d est valide\n",
 COMPTE, CONTROLE);
 else
     printf ("Le nombre CCP %ld-%d n'est pas valide\n",
 COMPTE, CONTROLE);
}
```

Exercice 11.16

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  /* Prototype de la fonction FUSION */
  void FUSION(int *A, int *B, int *FUS, int N, int M);
   /* Déclarations : */
  /* Noms des fichiers et pointeurs de référence */
  char FICH_A[30], FICH_B[30], FICH_FUS[30];
  FILE *FA, *FB, *FFUS;
  /* Autres variables */
  int *TABA, *TABB, *TFUS; /* pointeurs pour les tableaux */
  int LA, LB; /* Longueurs de FA et FB */
   int N; /* Nombre lu ou écrit dans un fichier */
  int I; /* Indice d'aide */

  /* Ouverture du fichier FA en lecture */
   do
    {
     printf("Nom du fichier FA : ");
     scanf("%s", FICH_A);
     FA = fopen(FICH_A, "r");
     if (!FA)
         printf("\aERREUR: Impossible d'ouvrir "
                "le fichier: %s.\n", FICH_A);
    }
```

```
while (!FA);
/* Détection de la longueur de FA */
for (LA=0; !feof(FA); LA++)
     fscanf(FA,"%d\n", &N);
/* Fermeture du fichier FA */
fclose(FA);
/* Allocation de la mémoire pour TABA */
TABA = malloc (LA*sizeof(int));
 if (!TABA)
     {
      printf("\a Pas assez de mémoire pour TABA\n");
      exit(-1);
     }


/* Réouverture du fichier FA en lecture */
FA = fopen(FICH_A, "r");
/* Copie du contenu de FA dans TABA */
for (I=0; I<LA; I++)
     fscanf(FA,"%d\n", TABA+I);
/* Fermeture du fichier FA */
fclose(FA);

/* Mêmes opérations pour FB : */
/* Ouverture du fichier FB en lecture */
 do
  {
   printf("Nom du fichier FB : ");
   scanf("%s", FICH_B);
   FB = fopen(FICH_B, "r");
   if (!FB)
       printf("\aERREUR: Impossible d'ouvrir "
              "le fichier: %s.\n", FICH_B);
  }
while (!FB);
/* Détection de la longueur de FB */
for (LB=0; !feof(FB); LB++)
     fscanf(FB,"%d\n", &N);
/* Fermeture du fichier FB */
fclose(FB);
/* Allocation de la mémoire pour TABB */
TABB = malloc (LB*sizeof(int));
 if (!TABB)
     {
      printf("\a Pas assez de mémoire pour TABB\n");
      exit(-1);
     }
/* Réouverture du fichier FB en lecture */
FB = fopen(FICH_B, "r");
/* Copie du contenu de FB dans TABB */
for (I=0; I<LB; I++)
```

```
      fscanf(FB,"%d\n", TABB+I);
  /* Fermeture du fichier FB */
  fclose(FB);

  /* Allocation de la mémoire pour TFUS */
   TFUS = malloc ((LA+LB)*sizeof(int));
  if (!TFUS)
      {
       printf("\a Pas assez de mémoire pour TFUS\n");
       exit(-1);
      }

  /* Fusion des tableaux TA et TB dans TFUS */
  FUSION (TABA, TABB, TFUS, LA, LB);


  /* Ouverture du fichier FFUS en écriture */
   do
    {
     printf("Nom du fichier FFUS : ");
     scanf("%s", FICH_FUS);
     FFUS = fopen(FICH_FUS, "w");
     if (!FFUS)
         printf("\aERREUR: Impossible d'ouvrir "
                "le fichier: %s.\n", FICH_FUS);
    }
  while (!FFUS);
  /* Copier le contenu de TFUS dans FFUS */
  for (I=0; I<(LA+LB); I++)
      fprintf(FFUS,"%d\n", *(TFUS+I));
  /* Fermeture du fichier FFUS */
  fclose(FFUS);
   return 0;
}

void FUSION(int *A, int *B, int *FUS, int N, int M)
{
  /* Variables locales */
 /* Indices courants dans A, B et FUS */
 int IA,IB,IFUS;

 /* Fusion de A et B dans FUS */
 IA=0, IB=0; IFUS=0;
 while ((IA<N) && (IB<M))
    if (*(A+IA)<*(B+IB))
        {
         *(FUS+IFUS)=*(A+IA);
         IFUS++;
         IA++;
        }
    else
```

```
        {
         FUS[IFUS]=B[IB];
         IFUS++;
         IB++;
        }
 /* Si A ou B sont arrivés à la fin, alors */
 /* copier le reste de l'autre tableau.    */
 while (IA<N)
     {
      *(FUS+IFUS)=*(A+IA);
      IFUS++;
      IA++;
     }
 while (IB<M)
     {
      *(FUS+IFUS)=*(B+IB);
      IFUS++;
      IB++;
     }
}
```