B Corrigés des exercices

```
/*********************************************************************/
/**** EXERCICE 1 : Comparer 2 entiers A et B quelconques et afficher ****/
/**** un message annonçant le résultat                               ****/
/*********************************************************************/

/****** L'algorithme ******
 *** DEBUT
 *** A, B : entier
 *** lire(A)
 *** lire(B)
 *** SI A<B ALORS
 ***     ecrire("A<B")
 *** SINON
 ***     SI A>B ALORS
 ***         ecrire("A>B")
 ***     SINON
 ***         ecrire("A=B")
 *** FIN
**************************/

#include <stdio.h>

int main (void)
{ int A, B;

 /* Lecture de A */
  printf("\n Valeur du nombre A : ");
  scanf("%d", &A);

 /* Lecture de B */
  printf("\n Valeur du nombre B : ");
  scanf("%d", &B);

  /* rappel des valeurs respectives de A et */
  printf("\n\nA = %d et B = %d", A, B);

  /* Comparaison et affichage du message correspondant */
  if (A>B)
      printf("--> A est plus grand que B\n");
  else  if (A==B)
           printf("--> A et B sont identiques\n");
        else
           printf("--> A est plus petit que B\n");
}

/*********************************************************************/
/**** EXERCICE 2 : Lire 2 valeurs entières en les plaçant dans deux  ****/
/**** variables x et y. Echanger les valeurs entre x et y            ****/
/****      puis afficher les valeurs de x et y.                     ****/
/*********************************************************************/

/****** L'algorithme ******
 *** DEBUT
 *** x, y, aux : entier
 *** lire(x)
```

```
 *** lire(y)
 *** aux <- x
 *** x <- y
 *** y <- aux
 *** ecrire(x)
 *** ecrire(y)
 *** FIN
**************************/

#include <stdio.h>

int main (void)
{  int x, y ;
   int aux; /* variable auxilliaire */

  /* Lecture de x */
  printf("\n Valeur du nombre x : ");
  scanf("%d", &x);

  /* Lecture de y */
  printf("\n Valeur du nombre y : ");
  scanf("%d", &y);

  /* echange des valeurs */
  aux = x ;
  x = y ;
  y = aux ;

  /* affichage */
  printf("x = %d \n",x);
  printf("y = %d \n",y);

}

/****** Autres algorithmes possibles******
 *** DEBUT
 *** x, y : entier
 *** lire(x)
 *** lire(y)
 *** x <- x-y
 *** y <- x+y
 *** x <- y-x
 *** ecrire(x)
 *** ecrire(y)
 *** FIN

 *** ou bien
 *** DEBUT
 *** x, y : entier
 *** lire(x)
 *** lire(y)
 *** x <- x+y
 *** y <- x-y
 *** x <- x-y
 *** ecrire(x)
 *** ecrire(y)
```

```
 *** FIN
**************************/

/*************************************************************************/
/**** EXERCICE 3 : Lire 3 valeurs entières en les plaçant dans 3     ****/
/**** variables x, y, z. Trier ces 3 valeurs de telle sorte que,     ****/
/**** au final, la plus petite des 3 valeurs soit placée dans x,     ****/
/**** la plus grande dans z et la valeur intermédiaire dans y.       ****/
/*************************************************************************/

/****** L'algorithme : raffinage 1 ******
 *** DEBUT
 *** x, y, z : entier
 *** lire(x)
 *** lire(y)
 *** lire(z)
 *** SI x>y ALORS
 ***    echanger x avec y
 *** SI y>z ALORS
 ***    echanger y avec z
 *** SI x>y ALORS
 ***    echanger x avec y
 *** ecrire(x)
 *** ecrire(y)
 *** ecrire(z)
 *** FIN
**************************/

/****** L'algorithme : raffinage 2 ******
 *** DEBUT
 *** x, y, z, aux : entier
 *** lire(x)
 *** lire(y)
 *** lire(z)
 *** SI x>y ALORS
 ***    DEBUT
 ***    aux <- x
 ***    x <- y
 ***    y <- aux
 ***    FIN
 *** SI y>z ALORS
 ***    DEBUT
 ***    aux <- z
 ***    z <- y
 ***    y <- aux
 ***    FIN
 *** SI x>y ALORS
 ***    DEBUT
 ***    aux <- x
 ***    x <- y
 ***    y <- aux
 ***    FIN
 *** ecrire(x)
 *** ecrire(y)
 *** ecrire(z)
 *** FIN
**************************/
```

```c
#include <stdio.h>

int main (void)
{  int x, y, z ;
   int aux; /* variable auxilliaire */

  /* Lecture de x, y et z */
  printf("\n Valeur des nombres x, y, z : ");
  scanf("%d", &x);
  scanf("%d", &y);
  scanf("%d", &z);

  /* tri */
  if (x>y) { aux = x ; x = y; y = aux ; }
  if (y>z) { aux = y ; y = z; z = aux ; }
  if (x>y) { aux = x ; x = y; y = aux ; }

  /* affichage */
  printf("x = %d, y =  %d, z = %d \n",x,y,z);
}

/*************************************************************************/
/**** EXERCICE 4 : Traduire la boucle for en une boucle while        ****/
/****      remarque : dans l'expression  (C = 0) && (I = 1)          ****/
/****      la partie (I=1) ne sera jamais évaluée contrairement      ****/
/****      à ce qui se passe avec la boucle                          ****/
/****      for (((C = 0), (I = 1)); (I > 0) ; I--) C+=I ;            ****/
/****                                                                ****/
/**** Rq : les opérateurs , et && sont évalués de gauche à droite    ****/
/****    Avec la , tout est évalué et la valeur de l'expression      ****/
/****    correspond à la dernière partie évaluée                     ****/
/****    Et avec && l'évaluation se poursuit tant qu'on ne trouve    ****/
/****    pas faux                                                    ****/
/**** Attention : le ; n'est pas un opérateur                        ****/
/*************************************************************************/
#include <stdio.h>

int main (void)
{  int C = 0, I = 1 ; // on pourrait donc ne pas mettre I=1; (cf remarque ci-dessus)
   while (I > 0) {
      C+=I ;
      I-- ;
   }
}

/*************************************************************************/
/**** EXERCICE 5 : Traduire la boucle while en une boucle for        ****/
/*************************************************************************/
#include <stdio.h>

int main (void)
{  int X ;
   float Y ;
   for (((X = 0),(Y = 0.25)) ; ((X == 0) || (Y != 2.)) ; Y = Y*2) ;
   // la virgule dans la première expression permet une évaluation complète
   // cela corrige le problème du && vu dans l'exo 4
```

```
}

/************************************************************************/
/**** EXERCICE 6 : Calculer la factorielle de 10.                    ****/
/************************************************************************/

/****** L'algorithme  ******
 *** DEBUT
 *** i, res : entier
 *** res <- 1
 *** POUR i DE 2 A 10 PARPASDE 1
 ***    res <- res * i
 *** ecrire(res)
 *** FIN
**************************/

#include <stdio.h>

int main (void)
{  int i, res = 1 ;
   for(i = 2; i <= 10; i++) res = res * i ;
   printf("factorielle de 10 = %d \n",res);
}

/************************************************************************/
/**** EXERCICE 7 : Déterminer le minimum et le maximum d'une série de ***/
/****              N nombres entiers  (N>0)                          ***/
/************************************************************************/
#include <stdio.h>

int main (void)
{ int I, N, MIN, MAX, NB;

 /* Lecture du nombre d'elements a traiter */
  printf("\n Combien de nombres voulez-vous traiter ? : ");
  scanf("%d", &N);

 /* Saisie du premier nombre et initialisation de MIN et MAX */
  printf("\nCommencer la saisie des %d nombres\n", N);
  scanf("%d", &NB);
  MIN = NB;
  MAX = NB;

 /* Boucle de traitement : chaque nombre lu est compare a MAX puis */
 /* a MIN s'il ne correspond pas au nouveau MAX                    */
  for( I = 1; I < N; I++)
  { scanf("%d", &NB);
    if (NB > MAX)
       MAX = NB;
    else /* NB n'est pas plus grand que le MAX, on le compare au min */
       if (NB < MIN) MIN = NB;
  }

 /* Ecriture du resultat */
  printf("\nValeur minimale = %d et valeur maximale = %d\n\n", MIN, MAX);

}
```

```
/*********************************************************************/
/**** EXERCICE 8 : version 1 :                                    ****/
/****              Calculer la moyenne d'une serie de N nombres  ****/
/****              entiers positifs en demandant le nombre       ****/
/****              d'elements a traiter a l'utilisateur          ****/
/****                                                            ****/
/**** Attention : ds la solution donnée ici, on ne teste pas le  ****/
/**** fait que NB soit positif ou non. On suppose que l'utilisateur ****/
/**** donne de bonnes valeurs                                    ****/
/**** On pourrait aussi definir NB comme unsigned int et le lire ****/
/**** avec le format %u                                          ****/
/*********************************************************************/
#include <stdio.h>

int main (void)
{ int I, N, NB;
  int SOMME = 0;          /* doit être absolument initialisé */
  float MOY ;

 /* Lecture du nombre d'elements a traiter */
  printf("\n Combien de nombres voulez-vous traiter ? : ");
  scanf("%d", &N);

  if(N > 0)
  { /* Saisie des nombres et calcul de la somme */
    printf("\nCommencer la saisie des %d nombres\n", N);
    for( I = 1; I <= N; I++)
    { scanf("%d", &NB);
      SOMME = SOMME + NB;               /* peut être ecrit SOMME += NB */
    }
    /* Calcul de la moyenne */
    MOY = (float) SOMME / (float) N;
                                  /* conversion obligatoire car SOMME   */
                                  /* et N etant des entiers la division */
                                  /* entiere serait effectuée entrainant*/
                                  /* la perte de la partie decimale     */

    /* Ecriture du resultat avec une partie decimale sur 6 chiffres, */
    /* puis arrondie à 2 chiffres                                    */
    printf("\n\n Moyenne obtenue : %f (arrondie a %.2f)\n\n", MOY, MOY);
 }
 else
  printf("le traitement ne peut pas être effectué\n");

}

/*********************************************************************/
/**** EXERCICE 8 : version 2                                      ****/
/****              Calculer la moyenne d'une serie de N nombres  ****/
/****              entiers positifs en arretant la saisie des que****/
/****              la valeur -1 est lue                          ****/
/**** -1 etant la condition d'arret, on ne peut pas definit NB comme ****/
/**** unsigned int, il faut tester qu'il soit positif pour en tenir  ****/
/**** compte                                                     ****/
/*********************************************************************/
#include <stdio.h>
```

```
int main (void)
{ int I, N, NB;
  int SOMME = 0;
  float MOY ;

 /* Saisie des nombres et calcul de la somme */
  printf("\nCommencer la saisie des nombres entiers >=0 \n(-1)termine la saisie\n");
  N = 0;
  scanf("%d", &NB);
  while ( NB != -1)
  { if (NB >=0)
    { SOMME += NB;  /* on ne prend en compte NB    */
                    /* que s'il est positif ou nul */
      N++;
    }
    scanf("%d", &NB);
  }

 /* Calcul de la moyenne */
  if (N != 0)
  {
      MOY = (float) SOMME / (float) N;

      /* Écriture du résultat */
      printf("\n\n Moyenne obtenue : %f (arrondie a %.2f)\n\n", MOY, MOY);
  }
  else
      printf("\n\n Pas de note ==> pas de moyenne ! \n\n");
}


/*************************************************************************/
/**** EXERCICE 9 : Calculer la somme des N premiers nombres impairs  ****/
/****                                                               ****/
/**** Partant de 1 qui est le premier nombre impair, compter le      ****/
/**** nombre de nombres impairs cumulés dans SOMME. Lorsque la valeur****/
/**** correspondant à N est atteinte, on s'arrête                    ****/
/**** Pour passer d'un nombre impair au suivant il suffit d'ajouter 2****/
/*************************************************************************/
#include <stdio.h>

int main (void)
{ int I, N, NB;
  int SOMME = 0;

 /* Saisie de la valeur de N --> nombre de nombres impairs a afficher */
  printf("\n\nValeur de N : ");
  scanf("%d", &N);

 /* Calcul de la somme des N premiers nombres impairs */
 NB = 1;
 for ( I =1; I <=N; I++)
 { printf("%d  ", NB);
   SOMME = SOMME + NB;
   NB +=2;
 }
```

```
/* Ecriture du resultat */
  printf("\n\n Somme obtenue : %d\n\n", SOMME);

}


/** Remarque : la somme de N nombre impairs est egale au carre de N */

#include<stdio.h>
#include<math.h>  /* nécessaire pour utiliser la fonction puissance pow*/
                  /* de plus il faut aussi compiler avec l'option -lm  */
                  /* cc prog.c -lm                                     */

void main(void)
{ int N;

 /* Saisie de la valeur de N --> nombre de nombres impairs a afficher */
  printf("\n\nValeur de N : ");
  scanf("%d", &N);

/* Écriture du résultat */
  printf("\n\n Somme obtenue : %d\n\n", (int) pow((double)N, (double)N) );

/* les fonctions de la bibliothèque mathématique math.h manipulent des  */
/* nombres flottants double précision, il faut donc procéder aux bonnes */
/* conversions pour ne pas avoir des résultats fausses                  */
}

/*************************************************************************/
/**** EXERCICE 10 : Déterminer si un nombre entier positif N est     ****/
/****              un nombre premier ou pas                          ****/
/*************************************************************************/
#include <stdio.h>

int main (void)
{ int I, N, NB;
  int premier_ok = 1; /* utilisation de premier_ok comme un booléen */
                      /* premier_ok initialise a 1 (=vrai) suppose  */
                      /* qu'on fait l'hypothèse que tout nombre est */
                      /* premier et qu'il faut prouver le contraire */
                      /* en trouvant un diviseur autre que 1 et lui */

 /* Saisie de la valeur de N --> nombre positif a traiter */
do
{
  printf("\n\nValeur de N > 0: ");
  scanf("%d", &N);
} while (N<=0);

/* on s'est assuré que N est bien positif et non nul */

/* Recherche d'un diviseur autre que 1 : partir de 2             */
/* Une optimisation possible : si 2 est un diviseur N/2 aussi.   */
/* Inutile d'aller au delà, des cas auront  été traités          */
/* ATTENTION : cela ne couvre pas le cas de N = 4                */
/*             => idée abandonnée                                */
/* Une autre optimisation en temps mais pas en espace : ne tester */
```

```
/* comme diviseur que les nb premiers inférieurs ou égaux à N/2   */
/*             => nécessité de les sauvegarder au fur et à mesure */
/*                (pas fait ici)                                   */

 for ( NB = 2; (NB <= (N/2)) && premier_ok; NB++)
 { if ((N % NB) == 0)
      premier_ok = 0;     /* on a un diviseur, arrêt du traitement */
 }

/* on utilise un for pour effectuer un nombre déterminé de fois      */
/* la meme action avec la possibilité de s'arrêter dès que l'on aura */
/* trouvé un diviseur. C'est suffisant pour prouver qu'un nombre     */
/* n'est pas premier                                                 */

 /* Ecriture du resultat */
 if (premier_ok)
    printf("\n\n %d est un nombre premier\n", N);
 else
    printf("\n\n %d n'est pas un nombre premier\n", N);

}

/*********************************************************************/
/**** EXERCICE 10 : Afficher les N premiers nombres qui sont des    ****/
/***                nombres premiers.                              ****/
/****                Il faut examiner tous les nombres, determiner  ****/
/****                s'ils sont premiers et si oui les comptabiliser ****/
/****                puis les afficher                              ****/
/*********************************************************************/
#include <stdio.h>

int main (void)
{ int IND, N, NB, nb_courant;
  int premier_ok = 1;

 /* Saisie de la valeur de N --> nombre de nombres a afficher */
do
{ printf("\n\nValeur de N > 0: ");
  scanf("%d", &N);
} while (N <= 0);

 nb_courant = 1;
 IND = 1;
 while (IND <= N)
 { /* Le nombre courant est-il premier               */
   /* on applique la methode de l'exercice precedent */
   premier_ok = 1;
   for ( NB = 2; (NB <= (nb_courant/2)) && premier_ok; NB++)
   { if ((nb_courant % NB) == 0)
        premier_ok = 0;
   }
   /* nombre courant est nombre premier --> affichage de ce nombre */
   if (premier_ok)
      { printf("%d   ", nb_courant);
        IND++;
      }
   /* nombre suivant */
```

```
   nb_courant++;
 }

}

/*************************************************************************/
/****  EXERCICE 11 : Trier une suite de N nombres entiers (N <= 20)  ****/
/****                suivant leur parite. Afficher d'abord les nombres****/
/****                pairs puis les nombres impairs                   ****/
/****                                                                 ****/
/**** Il est necessaire de memoriser les nombres pairs et les nombres****/
/**** impairs avant affichage. 2 tableaux de 20 elements un pour     ****/
/**** chaque catégorie de nombres. On peut faire le test de parite   ****/
/**** apres la lecture et memoriser le nombre directement dans le bon****/
/**** tableau                                                         ****/
/*************************************************************************/


#include <stdio.h>

int main (void)
{ int N, NB, I, IND_PAIR, NB_PAIR[20], IND_IMPAIR, NB_IMPAIR[20];

 /* Saisie du nombre d'elements a traiter */
do
{
  printf("Combien de nombres doit-on traiter (n <= 20)? : ");
  scanf("%d", &N);
} while ( (N<= 0) || (N > 20) );

 IND_PAIR = -1;
 IND_IMPAIR = -1;
 printf("\nCommencer la saisie des %d nombres\n", N);
 /* Boucle de traitement : lecture d'un nombre et test de parité */
 for( I = 0; I < N; I++)
 { scanf("%d", &NB);
   if ( (NB % 2) == 0)
   { /* NB est un nombre pair ; on le memorise dans NB_PAIR */
     IND_PAIR++;
     NB_PAIR[IND_PAIR] = NB;
   }
   else
   { /* NB est un nombre impair ; on le memorise dans NB_IMPAIR */
     IND_IMPAIR++;
     NB_IMPAIR[IND_IMPAIR] = NB;
   }
 }
 /* Ecriture du resultat du tri */

 if ( IND_PAIR == -1)
    printf("\nPas de nombre pair\n");
 else
    { printf("\nNombres pairs : \n");
      for ( I = 0; I <= IND_PAIR; I++)
          printf("%d  ", NB_PAIR[I]);
    }
```

```
 if ( IND_IMPAIR == -1)
    printf("\nPas de nombres impairs\n");
 else
    { printf("\nNombres impairs : \n");
      for ( I = 0; I <= IND_IMPAIR; I++)
         printf("%d  ", NB_IMPAIR[I]);
    }
}

/*************************************************************************/
/**** EXERCICE 12 : Calculer la moyenne des valeurs d'un tableau T  *****/
/**** dont l'indice est un multiple de 3.                           *****/
/*************************************************************************/
#include <stdio.h>

#define N 100               /* dimension du tableau  */

int main (void)
{  int T[N] ;
   int i, som = 0, nb = 0 ;
   float moy ;

   /* saisie du tableau */
   ...

   /* calcul de la moyenne */
   for (i=0; i < N; i+=3) {
       som = som + T[i] ;
       nb++ ;
   }
   moy = som / (float)nb ;  /* transtypage obligatoire */

   /* affichage resultat */
   ...
}

/*************************************************************************/
/**** EXERCICE 13 : Soit T1 un tableau d'entiers de dimension N.    *****/
/**** Calculer le tableau T2 de flottants de dimension N tel que    *****/
/**** chq case i de T2 contienne la moyenne des cases 0 à i de T1   *****/
/*************************************************************************/
#include <stdio.h>

#define N 100               /* dimension du tableau  */

int main (void)
{  int T1[N] ;
   float T2[N] ;
   int i, j, som ;

   /* saisie du tableau T1 */
   ...

   /* mise a jour de T2 */
   for (i=0; i < N; i++) {
       som = 0 ;
       for (j=0, j <= i ; j++) som = som + T1[j] ;
```

```
        T2[i] = som / (float)(i+1) ;  /* transtypage obligatoire */
    }

    /* affichage resultat */
    ...
}

/* Une autre version (plus efficace) */

int main (void)
{  int T1[N] ;
   float T2[N] ;
   int i, j, som = 0 ;

   /* saisie du tableau T1 */
   ...

   /* mise a jour de T2 */
   for (i=0; i < N; i++) {
       som = som + T1[i] ;
       T2[i] = som / (float)(i+1) ;  /* transtypage obligatoire */
   }

   /* affichage resultat */
   ...
}

/*************************************************************************/
/**** EXERCICE 14 : Soit T un tableau d'entiers de dimension N.     *****/
/**** Renverser les valeurs dans T. Version 1 : en utilisant un     *****/
/**** tableau auxilliaire                                           *****/
/*************************************************************************/
#include <stdio.h>

#define N 100               /* dimension du tableau  */

int main (void)
{  int T[N], Taux[N] ;
   int i ;

   /* saisie du tableau T */
   ...

   /* mise a jour de T en passant par Taux */
   for (i = 0 ; i < N; i++) Taux[i] = T[N-i-1] ;
   for (i = 0 ; i < N; i++) T[i] = Taux[i] ;

   /* affichage resultat */
   ...
}

/*************************************************************************/
/**** EXERCICE 14 : Soit T un tableau d'entiers de dimension N.     *****/
/**** Renverser les valeurs dans T. Version 2 : sans utiliser un    *****/
/**** tableau auxilliaire                                           *****/
/*************************************************************************/
#include <stdio.h>
```

```
#define N 100               /* dimension du tableau  */

int main (void)
{  int T[N] ;
   int i, aux ;

   /* saisie du tableau T */
   ...

   /* mise a jour de T en passant par aux */
   for (i = 0 ; i < N/2; i++) {
       aux = T[i] ;
       T[i] = T[N-i-1] ;
       T[N-i-1] = aux ;
   }

   /* affichage resultat */
   ...
}
```

```
/*********************************************************************/
/* EXERCICE 15 : Afficher la liste des lettres communes a 2 mots    ****/
/* Une lettre est commune si elle se trouve a la meme position dans ****/
/* les 2 mots                                                        ****/
/*********************************************************************/
#include <stdio.h>
#include <string.h>       /* bibliothèque de fonctions de manipulation */
                          /* des chaînes de caractères                 */

#define TRUE 1            /* définition de constantes symboliques pour */
#define FALSE 0           /* simuler l'utilisation de booléens         */

int main (void)
{ char mot1[26], mot2[26];  /* définition de 2 tableaux de caractères */
                            /* pour le stockage de 25 caractères max  */
                            /* + le marqueur de fin de chaîne \0 => 26*/
  char c;
  int lg1, lg2, lg, i;
  int lettre_ok = FALSE;      /* utilisation de lettre_ok comme booléen*/
                              /* on suppose au départ que les 2 mots   */
                              /* n'ont pas de lettres communes = FALSE */


  printf("\nPremier mot : ");
  scanf("%25s", mot1);            /* lecture de la première chaîne      */
                                  /* avec contrôle du nombre de lettres */

  while ( (c = getchar()) != '\n'); /* les caractères restants sont lus  */
                                    /* mais pas stockés => boucle de     */
                                    /* lecture caractère par caractère   */
                                    /* jusqu'au retour a la ligne        */
                                    /* ceci seulement si plus de         */
                                    /* caractères que nécessaire ont été */
                                    /* saisis par l'utilisateur          */

  lg1 = strlen(mot1);                  /* longueur de la première chaîne */
```

```
  printf("\nDeuxieme mot : ");         /* même contrôle pour le 2eme mot */
  scanf("%25s", mot2);
  while ( (c = getchar()) != '\n');
  lg2 = strlen(mot2);                   /* longueur de la seconde chaîne */


  lg = (lg1 < lg2)  ? lg1 : lg2;         /* longueur la plus courte */

  for( i=0; i < lg; i++)
  { if( mot1[i] == mot2[i])               /* meme lettre, meme position */
    { if (!lettre_ok)
      { /* premiere lettre commune trouvee              */
        /* affichage du message et de la lettre commune */
        printf("\nlettre(s) commune(s) : %c ", mot1[i]);
        lettre_ok = TRUE;
      }
      else
        printf("%c ", mot1[i]);        /* autres lettres communes */
    }
   }

   if(!lettre_ok)
     printf("\naucune lettre commune\n");
   else
     printf("\n");
}

/*********************************************************************/
/* EXERCICE 15 : Afficher la liste des lettres communes a 2 mots ****/
/* Une lettre est commune si elle se trouve dans chacun des mots ****/
/* quelle que soit sa position                                   ****/
/*********************************************************************/

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

#define TRUE 1
#define FALSE 0

int main (void)
{ char mot1[26], mot2[26] ;
  char let_com[26];            /* tableau supplémentaire pour    */
                               /* mémoriser les lettres communes */
  char c;
  int lg1, lg2, lg, i, nb_let =0;

  printf("\nPremier mot : ");
  scanf("%25s", mot1);
  while ( (c=getchar()) != '\n');
  lg1 = strlen(mot1);

  printf("\nDeuxieme mot : ");
  scanf("%25s", mot2);
  while ( (c=getchar()) != '\n');
  lg2 = strlen(mot2);
  for( i=0; i < lg1; i++)                  /* on parcourt le premier */
```

```
                                        /* mot lettre par lettre  */
  { if((strchr(mot2, mot1[i]) != NULL) &&
       (strchr(let_com, mot1[i]) == NULL))
    {   /* la lettre courante du premier mot se trouve dans le second  */
        /* mot et c'est la premier fois qu'elle apparaît puisqu'elle   */
        /* n'est pas deja dans le tableau des lettres communes         */
       let_com[nb_let] = mot1[i]; /* memorisation de la lettre commune */
       nb_let++;
    }
   }
   let_com[nb_let] = '\0';          /* ajout du marqueur de fin de chaine*/

   if(nb_let == 0)
     printf("\naucune lettre commune\n");
   else
     printf("lettre(s) commune(s) : %s", let_com);
}
```