Quelques exemples de programmes C

Bernard Cassagne

5 octobre 2012

Table des matières

```
/*****************************************************************************/
/*                                                                           */
/*                              essai1.c                                     */
/*                                                                           */
/*   utilisation des differents separateurs de lexemes                       */
/*                                                                           */
/*****************************************************************************/
int main()
{
int i;

/*   ici on a indente avec le caractere horizontal tabulation   */
        i = 0;
        i = 1;
        i = 2;

/*   ici la fin de ligne est utilisee comme separateur de lexeme   */
        i =
1;

/*  ici ce sont les commentaires qui sont separateurs de lexemes   */
   i  /*  bla bla  */ = /* bla bla  */ 1  ;

   return(0);
}
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai2.c                                       */
/*                                                                           */
/*   les identificateurs                                                     */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int ceci_est_un_identificateur;
int _identificateur_commencant_par_un_souligne;
int XXXXXXXX1;
int I;


int main()
{
  int XXXXXXXX2;   /* pour voir si ca va cacher la definition de XXXXXXXX1   */
  int i;           /* pour voir ce qui se passe avec I   */


  XXXXXXXX1 = 1;
  XXXXXXXX2 = 2;
  if (XXXXXXXX1 == XXXXXXXX2)
    printf("Au plus 8 caracteres significatifs\n");
  else printf("Plus de 8 caracteres significatifs\n");

  I = 1;
  i = 2;
  if (I == i)
    printf("On ne dispose que d'une seule fonte\n");
  else printf("On dispose des majuscules et des minuscules\n");

  return(0);
}


/*
------------------resultat de l'execution-------------------------------------

Au plus 8 caracteres significatifs
On dispose des majuscules et des minuscules
*/
```

```
/*************************************************************************/
/*                                                                       */
/*                            essai3.c                                   */
/*                                                                       */
/*     les constantes                                                    */
/*                                                                       */
/*************************************************************************/
#include <stdio.h>

int main()
{
  int i;
  long I;
  char c;
  float r;

  i = 8972;   /*   constante decimale   */
  i = 0344;   /*   constante octale   */

  /*   les constantes hexadecimales   */
  i = 0x7fb;
  i = 0x7FB;
  i = 0X7fb;
  i = 0X7FB;

  /*   les constantes longues   */
  I = 0l;
  I = 0L;
  I = 034l;
  I = 0x7abl;

  /*  les constantes caracteres   */
  c = 'a';
  c = '\n';   /*   line feed   */
  c = '\034';   /*   valeur du caractere en octal   */

  /*   chaines de caracteres   */
  printf("ceci est une chaine de caractere  ");
  printf("chaine\n\rsur trois\r\nlignes");
  printf("\033[12;40H positionnement du curseur sur le vt100");
  printf("chaine tenant\
sur plusieurs \
lignes de source");

  /*   les flottants   */
  r = 1.;
  r = .3;
  r = 1.3;
  r = 1e4;
  r = 1.e4;
  r = .3e4;
  r = 1.3e4;
  r = 1.3E4;
  r = 1.3E-4;
  r = 1.3e-4;

  return(0);

}
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai4.c                                     */
/*                                                                           */
/*    choix d'implementation concernant la longueur des types de base        */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int main()
{
  short int si;
  long int li;
  unsigned int ui;
  unsigned short us;
  unsigned long ul;

  printf("Taille des int : %d\n",sizeof(int));
  printf("Taille des char : %d\n",sizeof(char));
  printf("Taille des float : %d\n",sizeof(float));
  printf("Taille des double : %d\n",sizeof(double));

  printf("Taille des short int : %d\n",sizeof si);
  printf("Taille des long int : %d\n",sizeof li);
  printf("Taille des unsigned : %d\n", sizeof ui);
  printf("Taille des unsigned short : %d\n",sizeof us);
  printf("Taille des unsigned long : %d\n",sizeof ul);

  return(0);
}


/*
 ------------------------resultat de l'execution-----------------------------


Taille des int : 4
Taille des char : 1
Taille des float : 4
Taille des double : 8
Taille des short int : 2
Taille des long int : 4
Taille des unsigned : 4
Taille des unsigned short : 2
Taille des unsigned long : 4

*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai5.c                                     */
/*                                                                           */
/*     utilisation de l'operateur * (indirection)                            */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int t[10];   /*   tableau global   */

int *f(int i)   /*   procedure rendant un pointeur vers un int   */
{
  return(&t[i]);
}



int main()
{
  int *p;   /*   pointeur vers un int   */
  int i;

  /*   premier cas : la valeur pointeur est une variable   */
  /*   ------------------------------------------------   */
  i = 0;
  p = &i;
  *p = 2;
  printf("Valeur de i apres *p = 2 : %d\n",i);

  /*   deuxieme cas : la valeur pointeur est une expression arithmetique  */
  /*   ----------------------------------------------------------------  */
  p = &t[3];
  *(p + 1) = 6;
  printf("Valeur de t[4] apres p=&t[3]; *(p + 1) = 6 : %d\n",t[4]);

  /*   troisime cas : la valeur pointeur est resultat d'une fonction   */
  /*   ------------------------------------------------------------   */
  *f(5) = 7;
  printf("Valeur de t[5] apres *f(5) = 7 : %d\n",t[5]);

  return(0);
}


/*
------------------------resultat de l'execution-----------------------------


Valeur de i apres *p = 2 : 2
Valeur de t[4] apres p=&t[3]; *(p + 1) = 6 : 6
Valeur de t[5] apres *f(5) = 7 : 7
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai6.c                                     */
/*                                                                           */
/*   utilisation de l'operateur !                                            */
/*                                                                           */
/*****************************************************************************/

/*   include pour avoir la definition de NULL   */
#include <stdio.h>


int main()
{
  int *p1;   /*   pointeur vers un int   */
  int i;

  /*   ! applique a des expression aritmetiques   */
  /*   ---------------------------------------   */
  printf("!3 = %d\n",!3);
  printf("!(5 + 6) = %d\n",!(5 + 6));
  printf("!4.5e12 = %d\n",!4.5e12);
  printf("!0 = %d\n",!0);

  /*   ! applique a des pointeurs   */
  /*   --------------------------   */
  p1 = &i;
  i = !p1;
  if (i == (int)NULL) printf("!(&i) = NULL\n");

  p1 = NULL;
  printf("!NULL = %d\n",!p1);

  return(0);
}

/*
------------------------resultat de l'execution-----------------------------


!3 = 0
!(5 + 6) = 0
!4.5e12 = 0
!0 = 1
!(&i) = NULL
!NULL = 1

*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai7.c                                     */
/*                                                                           */
/*   utilisation de l'operateur ~                                            */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int main()
{
  int i,j;
  int size;

  size = sizeof(int);
  switch(size)
    {
    case 1 : i = 0xa5;
      break;
    case 2 : i = 0xa5a5;
      break;
    case 4 : i = 0xa5a5a5a5;
      break;
    default : printf("Taille des int = %d, non prevue par le programme\n",size);
      return(0);
    }

  j = ~i;
  printf("Valeur de j = %x\n",j);
  return(0);
}


/*
------------------------resultat de l'execution-----------------------------


Valeur de j = 5a5a5a5a
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai8.c                                       */
/*                                                                           */
/*   operateur sizeof                                                        */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int main()
{
  struct date
  {
    int jour;
    char mois[3];
    int annee;
  }  d;      /*   definition de la variable d de type struct   */

  int t[10];    /*   definition de la variable t tableau do 10 int   */

  typedef char LIGNE[100];   /*   definition du type LIGNE en tant   */
                             /*   que tableau de 100 char            */

  printf("Taille de d = %d\n",sizeof d);
  printf("Taille de date = %d\n",sizeof(struct date));
  printf("Taille de t = %d\n",sizeof t);
  printf("Taille de LIGNE = %d\n",sizeof(LIGNE));

  return(0);
}

/*
----------------------resultat de l'execution-----------------------------


Taille de d = 12
Taille de date = 12
Taille de t = 40
Taille de LIGNE = 100
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai9.c                                     */
/*                                                                           */
/*   operateurs ++ et --                                                     */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int main()
{
  int i;
  int t[10];
  int *p;

  i = 0;
  i++;   /*  incremente i  */
  printf("Valeur de i = %d\n",i);

  i = 0;
  t[i++] = 1;
  t[i++] = 2;

  i = 1;
  t[++i] = 3;
  t[++i] = 4;

  for (i = 0; i <= 3; i++)
    printf("t[%d] = %d ",i,t[i]);
  printf("\n");

/*   petit exemple sur la signification de *p++ :                           */
/*   ------------------------------------------                             */
/*   les operateurs * et ++ ont la meme precedence, mais l'associativite se  */
/*   fait de la droite vers la gauche meme si ++ est utilise en postfixe     */
/*   Donc, *p++ signifie *(p++)                                              */

  for (i = 0; i <= 9; t[i++] = 0) ;   /*   remise a zero du tableau t   */

  p = &t[0];
  (*p)++;   /*   incremente la valeur pointee   */

  p = &t[1];
  *p++ = 2;   /*  ++ incremente le pointeur   */

  *p = 3;

  printf("Valeur de t[0] = %d,t[1] = %d,t[2] = %d\n",t[0],t[1],t[2]);

  return(0);
}

/*
------------------------resultat de l'execution-----------------------------


Valeur de i = 1
t[0] = 1 t[1] = 2 t[2] = 3 t[3] = 4
Valeur de t[0] = 1,t[1] = 2,t[2] = 3
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai10.c                                    */
/*                                                                           */
/*   arithmetique sur les pointeurs                                          */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int main()
{
  int t[10];
  int *p1,*p2;
  int adr1, adr2;

  /*   exemple sur <pointeur> + <entier>   */
  /*   ----------------------------------   */
  p1 = &t[0];
  adr1 = (int) p1;    /*   on converti en int   */

  p1++;   /*   on incremente de 1 pour voir ce que ca va faire   */
  adr2 = (int) p1;   /*   on converti en int   */

  if (adr2 == adr1 + sizeof(int))
    printf("+1 sur le pointeur a incremente en fait de %d\n",sizeof(int));


  /*   exemple sur <pointeur> - <pointeur>   */
  /*   ------------------------------------   */
  p1 = &t[0];
  p2 = &t[9] + 1;   /*   pour pointer derriere le tableau   */
  printf("Nombre d'elements du tableau : %d\n",p2 - p1);

  return(0);
}

/*
------------------------resultat de l'execution-----------------------------


+1 sur le pointeur a incremente en fait de 4
Nombre d'elements du tableau : 10
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai11.c                                      */
/*                                                                           */
/*   operateurs & et && : appliqes a des booleen, ces 2 operateurs donnent   */
/*   le meme resultat. Leur difference reside dans le mode d'evaluation      */
/*   Le meme phenomene se produit entre | et ||                              */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

#define FALSE 0
#define TRUE  1

int effet_de_bord;   /*   variable de communication entre procedures   */

int fait_effet_de_bord()   /*   fonction retournant un int et faisant un
                               effet de bord                                */
{
  effet_de_bord = TRUE;
  return(TRUE);
}


int main()
{
  int i,j;

  printf("Table de verite de &\n\n");
  for (i = 0; i <= 1; i++)
    for (j = 0; j <= 1; j++)
      printf("%d & %d = %d\n",i,j,i & j);

  printf("\n\nTable de verite de &&\n\n");
  for (i = 0; i <= 1; i++)
    for (j = 0; j <= 1; j++)
      printf("%d && %d = %d\n",i,j,i && j);

  printf("\n\n");

  effet_de_bord = FALSE;
  i = 0 & fait_effet_de_bord();
  if (effet_de_bord)
    printf("& a evalue l'operande droit\n");
  else printf("& n'a pas evalue l'operande droit\n");

  effet_de_bord = FALSE;
  i = 0 && fait_effet_de_bord();
  if (effet_de_bord)
    printf("&& a evalue l'operande droit\n");
  else printf("&& n'a pas evalue l'operande droit\n");

  return(0);
}


/*
------------------------resultat de l'execution-----------------------------


Table de verite de &

0 & 0 = 0
0 & 1 = 0
1 & 0 = 0
1 & 1 = 1
```

```
Table de verite de &&

0 && 0 = 0
0 && 1 = 0
1 && 0 = 0
1 && 1 = 1


& a evalue l'operande droit
&& n'a pas evalue l'operande droit
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai12.c                                      */
/*                                                                           */
/*   operateur ?:                                                            */
/*   attention cet operateur a une associativite qui va de la droite vers    */
/*   la gauche                                                               */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int main()
{
  int i,j;

  /*   une utilisation de ?:   */
  /*   ----------------------  */
  i = 4;
  j = 5;
  printf("Le max de i et j est %c\n\n",i > j ? 'i' : 'j');

  /*   mise en evidence de l'associativite de l'operateur   */
  /*   ---------------------------------------------------  */
  printf("1 ? 1 : 0 ? 3 : 4 vaut %d\n",1 ? 1 : 0 ? 3 : 4);
  printf("(1 ? 1 : 0) ? 3 : 4 vaut %d\n",(1 ? 1 : 0) ? 3 : 4);
  printf("1 ? 1 : (0 ? 3 : 4) vaut %d\n",1 ? 1 : (0 ? 3 : 4));

  return(0);
}

/*
------------------------resultat de l'execution-----------------------------


Le max de i et j est j

1 ? 1 : 0 ? 3 : 4 vaut 1
(1 ? 1 : 0) ? 3 : 4 vaut 3
1 ? 1 : (0 ? 3 : 4) vaut 1
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai13.c                                      */
/*                                                                           */
/*   operateur virgule                                                       */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>
#include <string.h>

extern char last_car();   /* parcequ'elle est utilisee avant d'etre definie */
                          /* et qu'elle ne rend pas un int                  */

int main()
{
  static char s[] = "abcdefghijklmnopqrstuvwxyz";
  static char str1[] = "chaine avec line-feed\n";
  static char str2[] = "chaine sans lf";
  int i,j;

  /*   utilisation de l'operateur ',' dans un for      */
  /*   algorithme pour retourner une chaine en place   */
  /*   ---------------------------------------------   */
  for (i = 0, j = strlen(s) - 1; i < j; i++, j--)
    {
      char c;

      c = s[j];
      s[j] = s[i];
      s[i] = c;
    }

  printf("La chaine s est devenue : %s\n",s);


  /*   utilisation de l'operateur ',' ailleurs que dans un for   */
  /*   voir procedure last_car                                   */
  /*   ------------------------------------------------------    */
  printf("Dernier caractere de str1 : %c\n",last_car(str1));
  printf("Dernier caractere de str2 : %c\n",last_car(str2));

  return(0);
}


/*   procedure last_car rend le dernier caractere autre que line-feed d'une  */
/*   ligne pouvant etre ou ne pas etre terminee par line-feed                 */
char last_car(char s[])
{
  char c;
  int i;

  /*   remarquer le parenthesage, l' operateur ','
       a une priorite inferieure a celle de '='          */
  c = (i = strlen(s) - 1 , (s[i] == '\n') ? s[--i] : s[i]);
  return(c);
}


/*
------------------------resultat de l'execution-------------------------------


La chaine s est devenue : zyxwvutsrqponmlkjihgfedcba
Dernier caractere de str1 : d
Dernier caractere de str2 : f
```

```
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai14.c                                      */
/*                                                                           */
/*   variables pointeurs vers une fonctions                                  */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

/*   premiere partie : pointeurs vers des fonctions  */
/*   qui ne rendent pas une valeur                   */
/*   ----------------------------------------------  */
int f1()
{
  printf("Je suis la fonction f1\n");
  return(0);
}

int f2()
{
  printf("Je suis la fonction f2\n");
  return(0);
}

int test1()
{
  int (*pf)();   /*   la variable pointeur vers une fonction   */


  pf = f1;   /*   initialisation   */
  (*pf)();   /*   appel de la fonction   */

  pf = f2;
  (*pf)();

  return(0);

}


/*   deuxieme partie : pointeurs vers des fonctions   */
/*   qui rendent un pointeur vers un char             */
/*   -------------------------------------------      */
char s[10];

char *F1()
{
  return(&s[0]);
}

char *F2()
{
  return(&s[1]);
}

int test2()
{
  char *(*PF)();              /*  la variable pointeur vers une fonction    */
                              /*  qui rend un pointeur vers un char         */

  PF = F1;
  *(*F1)() = 'a';

  PF = F2;
  *(*F2)() = 'b';
```

```
  printf("La chaine s est devenue : %s\n",s);
  return(0);
}

int main()
{
  test1();
  test2();
  return(0);
}

/*
------------------------resultat de l'execution------------------------------


Je suis la fonction f1
Je suis la fonction f2
La chaine s est devenue : ab
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai15.c                                    */
/*                                                                           */
/*    tableaux de fonctions (en fait, de pointeurs vers des fonctions)       */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

/*   premiere partie : tableau de fonctions   */
/*   ---------------------------------------  */
int f1();   /*   il faut declarer les fonctions    */
int f2();   /*   car elles sont utilisees avant    */
int f3();   /*   d'etre definies                   */

int (*t_f[])() = {f1,f2,f3};  /* le tableau de pointeurs vers les fonctions  */

int f1()
{
  printf("Je suis la fonction f1\n");
  return(0);
}

int f2()
{
  printf("Je suis la fonction f2\n");
  return(0);
}

int f3()
{
  printf("Je suis la fonction f3\n");
  return(0);
}

int test1()
{
  (*t_f[0])();
  (*t_f[1])();
  (*t_f[2])();
  return(0);
}


/*   deuxieme partie : tableau de pointeurs vers des fonctions   */
/*   qui retournent un pointeur vers une structure               */
/*   ----------------------------------------------------------  */
struct date
{
  int jour;
  char mois[4];
  int annee;
} d1,d2;

struct date *F1()
{
  return(&d1);
}

struct date *F2()
{
  return(&d2);
}

struct date *(*T_F[])() = { F1,F2};  /*   le tableau de pointeurs   */
```

```
int test2()
{
  (*T_F[0])() -> jour = 1;
  (*T_F[1])() -> annee = 1984;
  printf("jour de d1 = %d annee de d2 = %d \n",d1.jour,d2.annee);
  return(0);
}

int main()
{
  test1();
  test2();
  return(0);
}

/*
------------------------resultat de l'execution-------------------------------


Je suis la fonction f1
Je suis la fonction f2
Je suis la fonction f3
jour de d1 = 1 annee de d2 = 1984
*/
```

```
/*************************************************************************/
/*                                                                       */
/*                            essai16.c                                  */
/*                                                                       */
/*    passage de parametre par adresse                                   */
/*                                                                       */
/*************************************************************************/
#include <stdio.h>

void echange(int *i,int *j)  /*   i et j sont des pointeurs   */
{
  int s;

  s = *i , *i = *j , *j = s;
}

int main()
{
  int a,b;

  a = 1 , b = 2;
  echange(&a,&b);   /*   on passe les adresses   */
  printf("Valeur de a = %d b = %d\n",a,b);

  return(0);
}

/*
-------------------------resultat de l'execution-------------------------------


Valeur de a = 2 b = 1
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai17.c                                    */
/*                                                                           */
/*   tableaux manipules en tant que tableaux ou avec un pointeur             */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>
#include <string.h>

/*   version de index ou on manipule un tableau   */
/*   ------------------------------------------   */
int index1(char s[],char c)
{
  int i,l;

  l = strlen(s);   /*   fonction de la bib standard   */
  for (i = 0; i < l; i++)
    if (s[i] == c) break;

  if (i < l) return(i); else return(-1);
}

/*   version de index ou on manipule un pointeur   */
/*   --------------------------------------------   */
int index2(  char *s,char c)
{
  int i,l;

  l = strlen(s);
  for (i = 0; i < l; i++)
    if (*s++ == c) break;   /* le parametre s est utilise comme   */
                            /* une variable locale                */

  if (i < l) return(i); else return(-1);
}

int main()
{
  static char str[] = "abcdefghijklmnopqrstuvwxyz";
  printf("Index de j = %d\n",index1(str,'j'));
  printf("Index de t = %d\n",index2(str,'t'));
  return(0);
}


/*
------------------------resultat de l'execution-------------------------------


Index de j = 9
Index de t = 19
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai18.c                                      */
/*                                                                           */
/*    passage de tableaux en parametre                                       */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

void add_vect(int v1[],int v2[],int v3[],int n)
     /*  on n'indique pas la dimension du tableau dans le   */
     /*  parametre formel, car c'est un pointeur qui sera   */
     /*  en fait passe en parametre                         */
     /*  il faut passer la dimension des tableaux en parametre       */
     /*  supplementaire, pour que la procedure puisse la connaitre   */
{
  int i;

  for (i = 0; i < n ; i++)
    v3[i] = v1[i] + v2[i];
}

int main()
{
  static int V1[3] = {1, 2, 3};
  static int V2[3] = {4, 5, 6};
  int V3[3];
  int i;

  add_vect(V1,V2,V3,3);
  printf("Vecteur V3 : ");
  for (i = 0; i < 3 ; i++) printf("%d ",V3[i]);
  return(0);
}


/*
------------------------resultat de l'execution-----------------------------


Vecteur V3 : 5  7  9

*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai19.c                                    */
/*                                                                           */
/*    tableaux de pointeurs - tableaux non carres                            */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

/*   premiere partie : tableau de pointeurs vers des types simples   */
/*   ------------------------------------------------------------   */
void test1()
{
  static int i = 1;
  static int j = 2;
  static int k = 3;
  static int *t[3] = {&i, &j, &k};
  /*   cette initialisation n'est acceptee que
       parceque les variables dont on prend les
       adresses sont "static"                  */

  printf("Valeurs pointees : %d  %d  %d\n",*t[0], *t[1], *t[2]);
}

/*   deuxieme partie : tableau de pointeurs vers des tableaux  */
/*   --------------------------------------------------------  */
void test2()
{
  static int t1[2] = {1, 2};
  static int t2[3] = {3, 4, 5};
  static int t3[4] = {6, 7, 8, 9};

  static int *t[3] = {t1,t2,t3};/*meme remarque que pour le tableau precedent*/
  static int l[3] = {2, 3, 4};  /*  indique les longueurs des tableaux   */

  int i,j;

  /*   on imprime le tableau t en accedant aux elements par t[i][j]   */
  printf("\nTableau t : \n");
  for (i = 0; i < 3; i++)
    {
      for (j = 0; j < l[i]; j++)
        printf("%d ",t[i][j]);
      printf("\n");
    }

  /* meme chose, en accedant aux elements de t par la notation *(t[i] + j)   */
  printf("\nTableau t : \n");
  for (i = 0; i < 3; i++)
    {
      for (j = 0; j < l[i]; j++)
        printf("%d ",*(t[i] + j));
      printf("\n");
    }
}

int main()
{
  test1();
  test2();
  return(0);
}


/*
------------------------resultat de l'execution-----------------------------
```

```
Valeurs pointees : 1  2  3

Tableau t :
1  2
3  4  5
6  7  8  9

Tableau t :
1  2
3  4  5
6  7  8  9

*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai20.c                                      */
/*                                                                           */
/*    tableaux a plusieurs dimensions passes en parametre                    */
/*    le but de ce programme est de voir les differences de programmation    */
/*    selon que l'on passe en parametre un vrai tableau a 2 dimensions,      */
/*    ou un tableau de pointeurs vers des tableaux uni-dimentionnels         */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

/*   premiere partie : on ecrit 3 versions d'un algorithme d'impression      */
/*   d'une matrice, le parametre etant un tableau a 2 dimensions             */
/*   ------------------------------------------------------------------      */
#define NC 2
void pr_mat1(int t[][NC],int nl)
     /*   il faut indiquer dans le type le nombre    */
     /*   d'elements de la seconde dimension si on   */
     /*   veut pouvoir utiliser la notation t[i][j]  */
     /*   L'algorithme n'est donc pas general.       */
     /*   n1 est le nombre d'elements de la premiere dimension      */

{
  int i,j;

  for (i = 0; i < nl; i++)
    {
      for (j = 0; j < NC; j++)
        printf("%d ",t[i][j]);
      printf("\n");
    }
}

void pr_mat2(int (*t)[NC],int nl)
     /*  il faut indiquer la taille des lignes pour que    */
     /*  l'instruction t++; fasse passer t d'une ligne     */
     /*  a la suivante. La aussi, l'algorithme n'est pas   */
     /*  general pour toutes les tailles de matrices       */

{
  int i,j;

  for (i = 0; i < nl; i++)
    {
      for (j = 0; j < NC; j++)
        printf("%d ",(*t)[j]);
      printf("\n");
      t++;                  /*   parametre t utilise comme variable cocale   */

    }
}


/*   maintenant on ecrit pr_mat3 de facon a accepter des matrices   */
/*   de n'importe quelle taille. Pour cela, le parametre est declare   */
/*   comme tableau unidimentionel, et on fait le calcul d'adresse      */
/*   'a la main' pour adresser t[i][j]                                 */
void pr_mat3(int t[],int nl,int nc)
{
  int i,j;

  for (i = 0; i < nl; i++)
    {
      for (j = 0; j < nc; j++)
        printf("%d ",t[i* nc + j]);
```

```
      printf("\n");
    }
}

void test1()
{
  static int t[3][2] = { {1, 2}, {3, 4}, {5, 6} };

  printf("Impression de t avec pr_mat1:\n");
  pr_mat1(t,3);

  printf("\nImpression de t avec pr_mat2 :\n");
  pr_mat2(t,3);

  printf("\nImpression de t avec pr_mat3 :\n");
  pr_mat3((int *)t,3,2);
}

/*   deuxieme partie : procedure acceptant une matrice sous forme d'un       */
/*   tableau de pointeurs vers des tableaux uni-dimentionnels de meme taille */
/*   L'algoritme est general et accepte la notation t[i][j]                  */
/*   ------------------------------------------------------------------------ */
void pr_mat(int *t[],int nl,int nc)
{
  int i,j;

  for (i = 0; i < nl; i++)
    {
      for (j = 0; j < nc; j++)
        printf("%d ",t[i][j]);
      printf("\n");
    }
}

void test2()
{
  static int t1[] = {10, 11};
  static int t2[] = {12, 13};
  static int t3[] = {14, 15};

  static int *T[] = {t1, t2, t3};

  printf("\nImpression de T avec pr_mat :\n");
  pr_mat(T,3,2);
}

int main()
{
  test1();
  test2();
  return(0);
}


/*
------------------------resultat de l'execution-----------------------------


Impression de t avec pr_mat1:
1  2
3  4
5  6

Impression de t avec pr_mat2 :
1  2
3  4
```

```
5  6

Impression de t avec pr_mat3 :
1  2
3  4
5  6

Impression de T avec pr_mat :
10  11
12  13
14  15
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai21.c                                    */
/*                                                                           */
/*   recuperation des arguments de la fonction main                          */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int main(int  argc, char *argv[])
{
  int i;

  for (i = 0; i < argc; i++)
    printf("argument numero : %d : %s\n",i,argv[i]);
  /* la procedure printf attend un pointeur vers une chaines de caracteres   */
  /* comme parametre de %s, c'est exactement ce qu'est argv[i]               */

  printf("Le quatrieme caractere du second 'vrai' argument est : %c\n",
         argv[2][4 - 1]);

  return(0);
}

/*
-----le programme est active avec la ligne de commande suivante :------------
essai21 '%()?.;;' fichier_employes -p 734



------------------------resultat de l'execution-------------------------------



argument numero : 0 : essai21
argument numero : 1 : %()?.;;
argument numero : 2 : fichier_employes
argument numero : 3 : -p
argument numero : 4 : 734
Le quatrieme caractere du second 'vrai' argument est : h
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai22                                      */
/*                                                                           */
/*    les structures                                                         */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>
#include <string.h>

int main()
{
  struct date
  {
    int jour;
    char mois[4];
    int annee;
  } d1,d2;

  struct date *p;

  /*   acces aux champs d'une variable   */
  /*   ------------------------------    */
  d1.jour = 1;
  strcpy(d1.mois,"jan");   /* impossible de faire d1.mois = "jan"        */
                           /* on utilise la fonction de copie de chaine  */
                           /*   de la bibliotheque standard              */

  d1.annee = 1984;

  /*   acces aux champs d'une structure pointee   */
  /*   ----------------------------------------   */
  p = &d2;
  p -> jour = 31;
  strcpy(p -> mois,"dec");   /*   meme remarque  */

  p -> annee = 1984;

  printf("d1 = %d %s %d \n",d1.jour,d1.mois,d1.annee);
  printf("d2 = %d %s %d \n",d2.jour,d2.mois,d2.annee);
  return(0);
}

/*
-------------------------resultat de l'execution-------------------------------


d1 = 1 jan 1984
d2 = 31 dec 1984
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai23                                        */
/*                                                                           */
/*    les structures et les fonctions : on ne peut pas passer en parametre   */
/*    une structure a une fonction, ni avoir une fonction qui retourne une   */
/*    structure.                                                             */
/*    Dans les 2 cas, il faut utiliser des pointeurs vers les structures.    */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

struct date
{
  int jour;
  char mois[4];   /*   trois caracteres significatifs seulement, a cause    */
                  /*   du \0 qui termine toute chaine                       */
  int annee;
};
struct date d1 = {1, "jan", 1984};

void pr_date(struct date *d)
{
  printf("%d %s %d\n",d -> jour, d -> mois, d -> annee);
}

struct date *give_date()
{
  return(&d1);
}

int main()
{
  give_date() -> annee = 2000;
  printf("Date modifiee :");
  pr_date(&d1);
  return(0);
}

/*
------------------------resultat de l'execution-----------------------------


Date modifiee :1 jan 2000
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai24.c                                    */
/*                                                                           */
/*    les tableaux de structures                                             */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>
#include <string.h>

struct date
{
  int jour;
  char mois[4];
  int annee;
};
struct date t[3] = { { 1, "avr", 1515}, { 14, "jui", 1789}, {1, "jan", 2000}};

void pr_date(struct date *d)   /*  imprime une date   */
{
  printf("%02d %s %d\n",d -> jour, d -> mois, d -> annee);
}

int main()
{
  int i;

  t[2].jour = 31;
  strcpy(t[2].mois,"dec");
  t[2].annee = 2001;

  printf("Tableau de dates :\n");
  for (i = 0; i < 3; i++) pr_date(&t[i]);
  return(0);
}


/*
------------------------resultat de l'execution-------------------------------


Tableau de dates :
01 avr 1515
14 jui 1789
31 dec 2001
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai25.c                                    */
/*                                                                           */
/*    E/S formattees : printf                                                */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int main()
{
  int i;
  char c;
  static char s[] = "chaine a imprimer";
  float f;

  /*   un entier en decimal   */
  printf("Un entier en decimal :\n");
  i = 134;
  printf("*%d*\n",i);   /*   s'imprime sur la place juste necessaire   */
  printf("*%6d*\n",i);  /*   s'imprime sur 6 caracteres                */
  printf("*%-6d*\n",i); /*   s'imprime sur 6 caracteres cadre a gauche */
  printf("*%06d*\n",i); /*   s'imprime avec des zeros devant           */

  /*  un entier en hexadecimal   */
  printf("\nUn entier en hexadecimal :\n");
  i = 0x6fe;
  printf("*%x*\n",i);
  printf("*%8x*\n",i);
  printf("*%-8x*\n",i);
  printf("*%08x*\n",i);

  /*   un entier en octal   */
  printf("\nUn entier en octal\n");
  i = 0377;
  printf("*%o*\n",i);
  printf("*%8o*\n",i);
  printf("*%-8o*\n",i);
  printf("*%08o*\n",i);

  /*   un unsigned imprime en decimal   */
  printf("\nUn unsigned imprime en decimal\n");
  i = -1;   /*   pour avoir le plus grand des unsigned   */
  printf("*%u*\n",i);
  printf("*%18u*\n",i);
  printf("*%-18u*\n",i);

  /*   un caractere   */
  c = 'a';
  printf("\nUn caractere : %c\n",c);

  /*   une chaine   */
  printf("\nUne chaine :\n");
  printf("*%s*\n",s);
  printf("*%25s*\n",s);
  printf("*%-25s*\n",s);
  printf("*%25.10s*\n",s);
  printf("*%-25.10s*\n",s);

  /*  un reel   */
  f = 1.7e+12;
  printf("\nUn reel :\n");
  printf("*%e*\n",f);
  printf("*%20e*\n",f);
  printf("*%-20e*\n",f);
  printf("*%20.7e*\n",f);
```

```
  printf("*%-20.7e*\n",f);

  return(0);
}

/*
------------------------resultat de l'execution-----------------------------

Un entier en decimal :
*134*
*   134*
*134   *
*000134*
*134   *

Un entier en hexadecimal :
*6fe*
*     6fe*
*6fe     *
*000006fe*
*6fe     *

Un entier en octal
*377*
*     377*
*377     *
*00000377*
*377     *

Un unsigned imprime en decimal
*4294967295*
*        4294967295*
*4294967295        *
*000000004294967295*
*4294967295        *

Un caractere : a

Une chaine :
*chaine a imprimer*
*        chaine a imprimer*
*chaine a imprimer        *
*               chaine a i*
*chaine a i               *

Un reel :
*1.700000e+12*
*        1.700000e+12*
*1.700000e+12        *
*       1.7000000e+12*
*1.7000000e+12       *

Un entier en decimal :
*134*
*   134*
*134   *
*000134*
*134000*

Un entier en hexadecimal :
*6fe*
*     6fe*
*6fe     *
*000006fe*
*6fe00000*
```

```
Un entier en octal
*377*
*     377*
*377     *
*00000377*
*37700000*

Un unsigned imprime en decimal
*4294967295*
*        4294967295*
*4294967295         *
*000000004294967295*
*4294967295000000000*

Un caractere : a

Une chaine :
*chaine a imprimer*
*        chaine a imprimer*
*chaine a imprimer         *
*               chaine a i*
*chaine a i                *

Un reel :
*1.70000e+012*
*        1.70000e+012*
*1.70000e+012         *
*      1.7000000e+012*
*1.7000000e+012       *

*/
```

```
/*************************************************************************/
/*                                                                       */
/*                            essai26.c                                  */
/*                                                                       */
/*   E/S formattees : scanf                                              */
/*                                                                       */
/*************************************************************************/
#include <stdio.h>

int main()
{
  int i,j,k;
  short l;
  char c1,c2,c3;
  char s[100];
  float f;

  printf("Entrez un entier en decimal:\n");
  scanf("%d",&i);

  printf("Entrez un entier en octal :\n");
  scanf("%o",&j);

  printf("Entrez un entier en hexadecimal :\n");
  scanf("%x",&k);

  printf("Entrez un entier decimal a mettre dans un short int :\n");
  scanf("%hd",&l);   /*   attention peut etre non-standard   */

  printf("Entrez 3 caracteres quelconqes contigus : \n");
  scanf("\n%c%c%c",&c1,&c2,&c3);  /*  le \n pour lire le line-feed de la reponse
                                      precedente; sinon c1 = line-feed     */


  printf("Entrez un flottant :\n");
  scanf("%f",&f);

  printf("Entrez une chaine de caracteres quelconques : \n");
  scanf("%s",s);

  printf("Les entiers lus sont :%d , %o , %x , %d\n",i,j,k,l);
  printf("Les 3 caracteres lus sont : '%c' , '%c' , '%c'\n",c1,c2,c3);
  printf("Le flottant est :%e\n",f);
  printf("La chaine lue est : **%s**\n",s);
  return(0);
}
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai27.c                                    */
/*                                                                           */
/*    E/S fichiers                                                           */
/*                                                                           */
/*****************************************************************************/

#include "stdio.h"

int main()
{
  FILE *fi,*fo;
  char buffer[256];

  fi = fopen("entree","r");
  if (fi == NULL)
    {
      printf("Impossible d'ouvrir le fichier entree\n");
      return(-1);
    }

  fo = fopen("sortie","w");   /*  le cree si il n'existe pas   */
  if (fo == NULL)
    {
      printf("Impossible d'ouvrir le fichier sortie");
      return(-1);
    }

  /*   on lit et on ecrit ligne a ligne   */
  while (fgets(buffer,256,fi) != NULL)
    fputs(buffer,fo);

  /*   on ferme les fichiers   */
  fclose(fi);
  fclose(fo);
  return(0);
}
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai28.c                                    */
/*                                                                           */
/*   les macros                                                              */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

#define champ_bit(x,i,n) (x) << ((i) - 1) >> (32 - (n))

int main()
{
  int i,j;

  i = 0x12345678;
  j = champ_bit(i,3,5);
  printf("Resultat = %x\n",j);

  return(0);
}


/*
-------------------------resultat de l'execution-------------------------------


Resultat = 9
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai29.c                                    */
/*                                                                           */
/*   Le pre-processeur                                                       */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int main()
{
#define OPTION
#ifdef OPTION
  printf("C'est cette ligne la qui va etre compilee\n");
#else
  printf("Celle-ci ne le sera pas\n");
#endif

#undef OPTION

#ifndef OPTION
  printf("Cette ligne aussi sera compilee\n");
#else
  printf("Celle ci non plus ne sera pas compilee\n");
#endif

#if 1 == 3 - 2
  printf("Cette ligne egalement sera compilee\n");
#else
  printf("C'est une erreur si cette ligne s'imprime\n");
#endif

  return(0);
}

/*
-------------------------resultat de l'execution-------------------------------


C'est cette ligne la qui va etre compilee
Cette ligne aussi sera compilee
Cette ligne egalement sera compilee
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai30.c                                    */
/*                                                                           */
/*    Fonction ayant un nombre variable de parametres                        */
/*                                                                           */
/*****************************************************************************/

#include "stdio.h"

void f(int *x,...)
{
  int **p;
  int i;

  p = &x;
  i = 0;
  while (*p++ != NULL) i++;
  printf("Nombre de parametres = %d\n",i);
}

 int main()
{
  int i,j,k;

  f(&i,NULL);
  f(&i,&j,NULL);
  f(&i,&j,&k,NULL);
  return(0);
}


/*
------------------------resultat de l'execution-----------------------------


Nombre de parametres = 1
Nombre de parametres = 2
Nombre de parametres = 3
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              esai3.1.c                                    */
/*                                                                           */
/*    types de base                                                          */
/*                                                                           */
/*****************************************************************************/
int i;
short int si;
long int li;

unsigned int ui;
unsigned short int usi;
unsigned long int uli;

short s;   /*   short synonyme de short int   */
long l;    /*   long synomyme de long int     */

unsigned u;          /*   unsigned synonyme de unsigned int   */
unsigned short us;   /*   unsigned short synonyme de unsigned short int   */
unsigned long ul;    /*   unsigned long synonyme de unsigned long int */

float f;
double d;

char c;
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai31.c                                    */
/*                                                                           */
/*   definition de types                                                     */
/*                                                                           */
/*****************************************************************************/

#include "stdio.h"   /*  pour avoir la definition de NULL   */

typedef struct noeud  /*  definition du type NOEUD   */
{
  int info;
  struct noeud *fils_gauche;
  struct noeud *fils_droit;
} NOEUD;

typedef NOEUD *P_NOEUD;  /*   definition du type pointeur vers NOEUD  */

NOEUD t_noeuds[100];
int i_noeud = 0;   /*  index dans le tableau t_noeuds   */

P_NOEUD alloc_noeud()   /*  alloue un NOEUD dans le tableau t_noeuds  */
{
if (i_noeud == 100)
   return(NULL);
 else
   {
     t_noeuds[i_noeud].info = 0;
     t_noeuds[i_noeud].fils_gauche = NULL;
     t_noeuds[i_noeud].fils_droit = NULL;
     return(&t_noeuds[i_noeud++]);
   }
}

void pr_arbre(P_NOEUD p)   /*   imprime un arbre en postfixe   */
{
  if (p == NULL)
    return;
  else
    {
      pr_arbre(p -> fils_gauche);
      pr_arbre(p -> fils_droit);
      printf("%d ",p -> info);
    }
}

int main()
{
  P_NOEUD p1,p2,p3;

  p1 = alloc_noeud();
  p2 = alloc_noeud();
  p3 = alloc_noeud();

  p1 -> fils_gauche = p2;
  p1 -> fils_droit = p3;
  p1 -> info = 32;
  p2 -> info = 65;
  p3 -> info = 72;

  pr_arbre(p1);
  return(0);
}
```

```
/*
----------------------resultat de l'execution-----------------------------


65  72  32
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai32.c                                    */
/*                                                                           */
/*   passage de fonctions en parametre                                       */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

int plus(int i,int j)
{
  return(i + j);
}

int mul(int i,int j)
{
  return(i * j);
}

int op_gen(int t[],int n, int (*operateur)())
     /* realise une operation sur les n elements du tableau*/
     /*   nombre d'elements du tableau   */
     /* pointeur vers la fonction realisant une operation  */
{
  int i,resu;

  switch (n)
    {
    case 0 : return(0);
    case 1 : return(t[0]);
    }
  resu = (*operateur)(t[0],t[1]);
  for (i = 2; i < n; i++)
    resu = (*operateur)(resu,t[i]);
  return(resu);
}

int main()
{
  static int t[5] = {1, 2, 3, 4, 5};
  int somme,produit;

  somme = op_gen(t,5,plus);
  printf("Somme des 5 premiers entiers = %d\n",somme);

  produit = op_gen(t,5,mul);
  printf("Produit des 5 premiers entiers = %d\n",produit);
  return(0);
}


/*
------------------------resultat de l'execution-----------------------------


Somme des 5 premiers entiers = 15
Produit des 5 premiers entiers = 120
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai33.c                                    */
/*                                                                           */
/*   structures passees en parametre ou retournees par une fonction          */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

struct complexe
{
  double x,y;
};

struct complexe *add_comp(struct complexe *c1,struct complexe *c2)
{
  static struct complexe r;

  r.x = c1 -> x + c2 -> x;
  r.y = c1 -> y + c2 -> y;

  return(&r);
}

int main()
{
  static struct complexe c1 = { 3.2, 4.5};
  static struct complexe c2 = { 6.5, 7.9};
  struct complexe *pc;

  pc = add_comp(&c1,&c2);
  printf("resu = %e %e\n",pc -> x, pc -> y);
  return(0);
}


/*
------------------------resultat de l'execution-----------------------------


resu = 9.70000e+000 1.24000e+001
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai34.c                                    */
/*                                                                           */
/*    E/S fichiers en utilisant getc et putc                                 */
/*                                                                           */
/*****************************************************************************/

#include "stdio.h"

int main()
{
  FILE *fi,*fo;
  int c;

  fi = fopen("entree","r");
  if (fi == NULL)
    {
      printf("Impossible d'ouvrir le fichier entree\n");
      return(-1);
    }

  fo = fopen("sortie","w");   /*  le cree si il n'existe pas   */
  if (fo == NULL)
    {
      printf("Impossible d'ouvrir le fichier sortie");
      return(-1);
    }

  /*   on lit et on ecrit ligne a ligne   */
  while (( c = getc(fi)) != EOF)
    putc(c,fo);

  /*   on ferme les fichiers   */
  fclose(fi);
  fclose(fo);
  return(0);
}
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai35.c                                      */
/*                                                                           */
/*     compilation separee (ce module doit etre lie avec essai36             */
/*                                                                           */
/*****************************************************************************/

extern int k;                       /*   variable importee du module 2    */
int i;                              /*   variable exportee par le module 1   */
static int j;                       /*   variable privee du module 1   */
int t[5] = {12, 34, 56, 78, 90};  /*  tableau exporte par le module 1  */

extern int f2();  /*   fonction importee du module 2   */
static int g();   /*   fonction privee du module 1   */

int f1(int imprime)        /*   fonction exportee par le module 1   */
{
  if (imprime) g();
  return(0);
}

static int g()   /*   fonction privee du module 1   */
{
  f2("i",i);
  f2("j",j);
  f2("k",k);
  return(0);
}
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai36.c                                    */
/*                                                                           */
/*    compilation separee (ce module doit etre lie avec essai35              */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

extern int i;     /*   on importe la variable i du module 1   */
static int j;     /*   variable privee du module 2     */
int k;            /*   variable exportee par le module  2   */
extern int t[];   /*   tableau importe du module 1   */

extern int f1();   /*   fonction importee du module  1   */

int f2(char *s,int i)
{
  printf("Valeur de %s = %d\n",s,i);
  return(0);
}

static int g()   /*   fonction privee du module  2   */
{
  i = 1;
  j = 2;
  k = 3;
  f1(1);    /*   appel de la procedure externe   */
  return(0);
}

int main()   /*  la procedure lancee par le systeme  */
{
  printf("Le tableau t du module 1 contient les valeurs suivantes: \n");
  printf("%d",t[0]);
  for ( i = 1; i <= 4; i++) printf(", %d",t[i]);
  printf("\n\n");
  g();
  return(0);
}

/*
------------------------resultat de l'execution-------------------------------
Le tableau t du module 1 contient les valeurs suivantes:
12, 34, 56, 78, 90

Valeur de i = 1
Valeur de j = 0
Valeur de k = 3
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                            essai37.c                                      */
/*                                                                           */
/*    utilisation de calloc et cfree                                         */
/*                                                                           */
/*****************************************************************************/
#include <stdlib.h>

struct noeud {
  int type_noeud;
  int info;
  struct noeud *fils_gauche;
  struct noeud *fils_droit;
} *p_n;

struct date {
  int jour;
  char mois[4];
  int annee;
} *p_d;

/* char *calloc();   calloc declaree comme rendant un pointeur vers un char
                       car calloc va rendre des pointeurs de type differend */

int main()
{
  p_n = (struct noeud *) calloc(100, sizeof(struct noeud));
  p_d = (struct date *) calloc(10,sizeof(struct date));

  cfree(p_n);
  cfree(p_d);
  return(0);
}
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai38.c                                    */
/*                                                                           */
/*   champs de bits                                                          */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>
#include <string.h>

#define MASCULIN 0
#define FEMININ 1

#define CELIBATAIRE  0
#define MARIE        1
#define DIVORCE      2
#define VEUF         3
#define CONCUBIN     4

#define OUI 0
#define NON 1

struct etat_civil {
  char nom[20];
  unsigned sexe : 1;
  unsigned situation : 3;
  unsigned invalide : 1;
  char adresse[50];
};

struct etat_civil e;

void pr_etat(struct etat_civil *p)
{
  static char *s[5] = { "celibataire", "marie","divorce","veuf","concubin"};

  printf("NOM : %s\n",p -> nom);

  printf("sexe : ");
  if (p -> sexe == MASCULIN)
    printf("masculin\n");
  else printf("feminin\n");

  printf("situation de famille : %s\n",s[p -> situation]);

  printf("adresse : %s\n",p -> adresse);
}


int main()
{
  e.sexe = FEMININ;
  e.situation = VEUF;
  e.invalide = NON;
  strcpy(e.nom,"Dupond");
  strcpy(e.adresse,"20 rue des lilas\n33200 ARCACHON");

  pr_etat(&e);
  return(0);
}

/*
------------------------resultat de l'execution-------------------------------


NOM : Dupond
```

```
sexe : feminin
situation de famille : veuf
adresse : 20 rue des lilas
33200 ARCACHON
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                           essai39.c                                       */
/*                                                                           */
/*   exemple sur les unions : on decrit les structures de donnees            */
/*   permettant de manipuler un arbre. Il y a deux sortes de noeuds :        */
/*   les noeuds operateurs et les noeuds feuilles. Ces derniers peuvent      */
/*   etre de deux types : les identificateurs et les nombres.                */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>

#define FEUILLE         1
#define OPERATEUR       2

#define IDENTIFICATEUR  1
#define NOMBRE          2

struct s_feuille
{
  int typ_feuille;   /*   peut valoir IDENTIFICATEUR ou NOMBRE   */
  union
  {
    char *ident;   /*   pour les identificateurs   */
    int valeur;    /*   pour les nombres   */
  }u;
};

struct s_oper
{
  char oper;   /*  l'operateur   */
  struct s_noeud *fils_g;   /*   le fils gauche   */
  struct s_noeud *fils_d;   /*   le fils droit    */
};

struct s_noeud
{
  int typ_noeud;   /*   peut valoir FEUILLE ou OPERATEUR   */
  union
  {
    struct s_feuille f;
    struct s_oper o;
  } u;
};

/*   define pour se simplifier considerablement l'acces aux champs   */
/*   ------------------------------------------------------------   */
#define TYP_FEUILLE     u.f.typ_feuille
#define IDENT           u.f.u.ident
#define VALEUR          u.f.u.valeur
#define OPER            u.o.oper
#define FILS_G          u.o.fils_g
#define FILS_D          u.o.fils_d

/*   fonction d'allocation d'un noeud feuille de type identificateur   */
/*   ---------------------------------------------------------------   */
struct s_noeud *alloc_idf(char *ident)
{
  struct s_noeud *p;

  p = (struct s_noeud *) malloc(sizeof(struct s_noeud));
  p -> typ_noeud = FEUILLE;
  p -> TYP_FEUILLE = IDENTIFICATEUR;
  p -> IDENT = ident;
  return(p);
```

```
}

/*   fonction d'allocation d'un noeud feuille de type nombre   */
/*   ------------------------------------------------------   */
struct s_noeud *alloc_nb(int val)
{
  struct s_noeud *p;

  p = (struct s_noeud *) malloc(sizeof(struct s_noeud));
  p -> typ_noeud = FEUILLE;
  p -> TYP_FEUILLE = NOMBRE;
  p -> VALEUR = val;
  return(p);
}

/*   fonction d'allocation d'un noeud de type operateur   */
/*   ---------------------------------------------------   */
struct s_noeud *alloc_oper(char oper,
                           struct s_noeud *fils_g,
                           struct s_noeud *fils_d)
{
  struct s_noeud *p;

  p = (struct s_noeud *) malloc(sizeof(struct s_noeud));
  p -> typ_noeud = OPERATEUR;
  p -> OPER = oper;
  p -> FILS_G = fils_g;
  p -> FILS_D = fils_d;
  return(p);
}

/*   procedure d'ecriture d'un arbre   */
/*   -------------------------------   */
void print_arbre(struct s_noeud *a)
{
  if (a -> typ_noeud == FEUILLE)
    {
      if (a -> TYP_FEUILLE == IDENTIFICATEUR)
        printf("%s",a -> IDENT);
      else printf("%d",a -> VALEUR);
    }
  else
    {
      print_arbre(a -> FILS_G);
      printf(" %c ",a -> OPER);
      print_arbre(a -> FILS_D);
    }
}

/*   programme principal   */
/*   --------------------   */
int main()
{
  struct s_noeud *n1,*n2,*n3,*n4,*n5;

  /*   on construit un arbre   */
  /*   ---------------------   */
  n1 = alloc_idf("i");
  n2 = alloc_nb(24);
  n3 = alloc_oper('+',n1,n2);
  n4 = alloc_idf("j");
  n5 = alloc_oper('=',n4,n3);

  /*   impression de l'arbre construit   */
  /*  -------------------------------   */
  printf("l'arbre est : ");
```

```
  print_arbre(n5);
  printf("\n");
  return(0);
}



/*
-----------------------resultat de l'execution-------------------------------


l'arbre est : j = i + 24
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai40                                      */
/*                                                                           */
/*   Fonctions rendant un pointeur vers une fonction                         */
/*   ATTENTION                                                               */
/*      1) il ne faut pas donner le type void comme valeur rendue par p1 et  */
/*         p2, sinon les return(p1) et return(p2) dans f() donnent une erreur */
/*      2) return(p1) et return(p2) dans f() generent un warning, mais le    */
/*         code genere est ok                                                */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

void p1()
{
  printf("je suis p1\n");
}


void p2()
{
  printf("je suis p2\n");
}


void (*f(int b))()
{
  if (b) return(p1); else return(p2);
}

int main()
{
  void (*pf)();

  pf = f(1);
  (*pf)();

  pf = f(0);
  (*pf)();

  (*f(1))();
  (*f(0))();
  return(0);
}

/*
-------------------------resultat de l'execution-------------------------------
je suis p1
je suis p2
je suis p1
je suis p2
*/
```

```
/*****************************************************************************/
/*                                                                           */
/*                              essai41                                      */
/*                                                                           */
/*   Fonctions rendant un pointeur vers une fonction                         */
/*                                                                           */
/*****************************************************************************/
#include <stdio.h>

struct rationnel { int x,y;} ;
typedef struct rationnel Rationnel;
typedef struct rationnel *PtrRationnel;
Rationnel R1 = {1,2};
Rationnel R2 = {3,4};

PtrRationnel p1()
{
  return(&R1);
}


PtrRationnel p2()
{
  return(&R2);
}


PtrRationnel (*f(int b))()
{
  if (b) return(p1); else return(p2);
}

int main()
{
  PtrRationnel p;

  p =  (*f(1))();
  printf("x = %d y = %d\n",p -> x, p-> y);

  p =  (*f(0))();
  printf("x = %d y = %d\n",p -> x, p-> y);
  return(0);
}


/*
------------------------resultat de l'execution-----------------------------
x = 1 y = 2
x = 3 y = 4

*/
```

```
/*****************************************************************************/
/**                                                                         **/
/** @(#) title: Essais avec les qualificateurs de type                    >**/
/** @(#) keywords:                                                        >**/
/**                                                                         **/
/*****************************************************************************/
const int ci;
const int *pci;
const int * const cpci;
int *p;
f1()
{
  *p = 1;

  /*   INTERDIT : ci constant int   */
#if 0
  ci = 1;
#endif

  pci = p;    /*   AUTORISE : pci n'est pas constant   */
  /*   INTERDIT : pci pointeur vers constant int   */
#if 0
  *pci = 1;
#endif

  /*   INTERDIT : cpci pointeur constant */
#if 0
  cpci = p;
#endif

}
/*****************************************************************************/
/*****************************************************************************/
struct s { int a, b;};
const struct s s1;
struct s s2;
struct S { const int a; int b;};
struct S s3;

f2()
{
  s2 = s1;

  /*   INTERDIT s2 structure constante   */
#if 0
  s1 = s2;    /*   modif de la structure entiere interdit   */
  s1.a = 1;   /*   modif d'un champ interdit   */
#endif

  /*   acces au seul champ a interdit   */
#if 0
  s3.a = 1;
#endif
  s3.b = 1;    /*   acces au champ b autorise   */
}
/*****************************************************************************/
/*****************************************************************************/
const t[10];

f3()
{
  /*   INTERDIT : tout element est const   */
#if 0
  t[0] = 0;
#endif
```

```
}
```

```c
/*****************************************************************************/
/**                                                                         **/
/** @(#) title: Comment allouer un tableau a deux dimensions par calloc     >**/
/** @(#) keywords:                                                          >**/
/**                                                                         **/
/*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>

#define NBLIG 10   /*   nombre de lignes   */
#define NBCOL 5    /*   nombre de colonnes   */
int main()
{
  int (*p)[][NBCOL];   /*   le type est pointeur vers tableau   */
  int i, j;
  int *q;

  /*   Allocation de la memoire pour le tableau   */
  /*   ---------------------------------------   */
  p =  (int (*)[][NBCOL]) calloc(NBLIG*NBCOL, sizeof(int));

  /*   On remplit le tableau   */
  /*   ---------------------   */
  for (i = 0; i <= NBLIG-1; i++)
    for (j = 0; j <= NBCOL-1; j++)
      (*p)[i][j] = 10*i + j;

  /*   Pour verifier qu'il a ete remplit comme on le pense on      */
  /*   le lit comme s'il etait un tableau a une seule dimension    */
  /*   ---------------------------------------------------------   */
  q = (int *)p;
  for (i = 0; i <= NBLIG*NBCOL-1 ; i++)
    printf("%d\n", *q++);
  return(0);
}
```