Université Lille Nord de France
Pôle de Recherche et d'Enseignement Supérieur

Université de Valenciennes et du Hainaut-Cambrésis

ALGORITHMIQUE ET PROGRAMMATION C NIVEAU 2

UVHC – ISTV – Licence 2

Rabie Ben Atitallah
rabie.benatitallah@univ-valenciennes.fr
http://www.lifl.fr/~benatita/pages/Teaching

CHAP III STRUCTURES ET POINTEURS

I Structures ou Enregistrements

type structure = struct [ident] [(décl. du champ)]

```
struct date{      short j, m ;
                  int a ;
           } /* solution 1 */
struct date d1, d2;
```

```
struct monome   {    int degre;
                     float coef;
                }
 struct monome  m, tab_monomes [ MAX];
```

Structures ou Enregistrements (suite)

```
struct date    {   short j, m ;    int a ; }
typedef struct date Typ_date; /*déclaration de type: sol 2*/
Typ_date  d1, d2 ;
```

```
typedef struct { short j, m ; int a; } Typ_date;
Typ_date  d1, d2;  /* déclaration de type: solution 3 */
```

```
typedef  enum { lun, mar, mer, jeu, ven, sam, dim}
   T_jour;
typedef  struct
      {     T_jour  jl ;
            Typ_date  d ;
      } Une_date  ;  /* un autre type  */
```

Opérateurs : affectation et sélection d'un champ

```
Une_date  auj, demain ;
auj . jl = mar ; auj . d . a = 2003 ; auj . d . m = 10 ;
demain = auj ;
```

```
void afficher  ( struct date d)
{   printf («%d  / %d / %d \n»,
            d.j, d.m. d.a  );
 }

void affich  (Une_date  x )
{  afficher ( x.d ) ;
   affich_jour ( x.jl ) ;
 }
```

```
struct date ld2( )
{    struct date d ;
     scanf  («%d%d%d », &(d.j),
                    &(d.m), &(d.a) ) ;
    return d  ;
}
Une_date  lire_d ( )
{   Une_date  x ; int  jour;
     x.d = ld2 ( ) ;
    scanf (« %d », & jour);
    x . jl= ( T_jour )  ( jour  % 7 ) ;
    return   x ;    }
```

II LES POINTEURS

II-1 Définition

Valeurs d'un pointeur = **adresses** d'autres objets.

Intérêt en C :

mécanisme d'indexation, paramètre résultat

allocation dynamique et type récursif (listes,...)

II-2 Déclaration des pointeurs

```
int *p          /* p  est pointeur sur des objets de type entier. */
enum jour *q ;      /* q est un pointeur sur un type enum */
int *p[N] ;  /*tableau p ayant N valeurs pointeurs d'entiers*/
int (*p)[N] ;   /* p est un pointeur sur 1 tableau de N entiers*/
```

II-3 Opérateur de prise d'adresses

Opérateur **unaire** **&** appliqué à une variable.

```
int   x = 1 ,  t[N],  *p , * q ;
{      p = &x ;   q =  t ;          }
```

p fait de ***p*** un pointeur et ***&x*** donne l'adresse de *x* .

II-4 Opérateur d'adresse indirecte

Le pointeur sert à faire des indirections et ainsi à obtenir des valeurs.

```
int    x  = 4 ,  y = 7,  *p = &x ;
{        *p = y ;                /* x  <-- y */
         *p = 0 ;                /* mettre x à 0 */
         *p + = y ;              /* x <--  x + y */
}
```

UN EXEMPLE

```
main ( )
{ int k = 3, *p = &k, q;
   printf ("adresse = %d et valeur = %d\n", p,
   *p);
  *p = 5;
  printf ( " adresse = %d et valeur = %d\n", p,
   *p);
  k=8;
  printf ("adresse = %d et valeur = %d\n", p,
   *p);
  *p = 2;
  printf (" adresse = %d et valeur = %d\n", &k,
   k);
  k = *p * k;
 printf (" adresse = %d et valeur = %d\n", p, k);
}
```

UN SECOND EXEMPLE

```
void lire_deux_reel ( float *pa,
  float * pb)
{ float a, b;
  printf (" entrer 2 valeurs  ");
  scanf ("%f%f", &a, &b);
  *pa = a;
  *pb = b;
}
```

```
void lire_deux_reel ( float *pa, float * pb)
{  printf ("entrer 2 valeurs ");
   scanf ("%f%f", pa, pb);
}
```

II-5 Opérateurs sur les pointeurs

- une valeur commune : ***NULL*** ; (ne pointe sur rien)
- affectation de pointeurs
- addition ou soustraction d'un entier
- soustraction de deux pointeurs
- comparaison de deux pointeurs (< , > , == , !=)

II-6 Pointeur universel

C'est un pointeur sur un type non déterminé :

void *pu ;

pu pointe sur des valeurs vides mais on pourra le forcer à pointer sur un autre type (forceur)

II-7 Pointeur et indexation

C'est une forme particulière de l'indirection :

expr1 [expr2] ;	****(expr1 + (expr2))***
t[i] ;	****(t + i)***
g[i][j] ;	****(*(g + i) + j)***

II-8 Pointeur et tableau

Le nom d'un tableau = adresse = pointeur

int j [3] , *q = & j [0] , *q1 ;

q est équivalent à ***j***

****q*** est équivalent à ***j [0] = *(&j[0])***

****(q + 0)*** est équivalent à ****q***

****j*** est équivalent à ****(j + 0)***

Pointeur et tableau sont syntaxiquement équivalents. En fait, le tableau est un pointeur constant.

Fonction qui donne la longueur d'une chaîne

entrée :c : chaîne

sortie : longueur : entier

```
int lng ( char *c )
{       int lg = 0 ;
        while ( *c !='0')
                {lg++;
                 c++;
                }
        return  lg  ;
}
```

```
int lng ( char c [ ] )
{   int lg = 0 ;
    while ( c[ lg] !='\0')
                    lg++;
     return   lg  ;
}
```

```
int lng ( char *c )
{       int lg = 0 ;
        while ( *c )
                {lg++;
                 c++;
                }
        return  lg  ;
}
```

```
int lng ( char c [ ] )
{   int lg = 0 ;
    while ( c[ lg] )
                    lg++;
     return   lg  ;
}
```

```
int lng ( char *c )
{       int lg = 0 ;
        while ( *c ++)  lg++;
        return  lg  ;
}
```

```
int max ( int t [ ] , int nb)
{      int i=0, rep = t [0];
       while (i++< nb-1)
              if ( t[i] > rep ) rep = t [i];
       return rep;
}
```

```
int max ( int t [ ] , int nb)
{      int i, rep = t [0];
       for (i=0; i< nb; i++)
              if ( t[i] > rep ) rep = t [i];
       return rep;
}
```

```
int max ( int  * t  , int nb)
{      int i = 0, rep = *t ;
       while (i++< nb-1)
            if (*(t+i)>rep ) rep=*(t+i);
       return rep;
}
```

```
int max ( int  * t  , int nb)
{      int  rep = *t ;
       while ( - - nb > 0)
              { if (*t>rep) rep=*t; t++; }
       return rep;
}
```

II-9 Pointeurs vers les structures

On peut transmettre à une fonction l'adresse d'une structure. L'opérateur **&** transmet l'adresse de la structure ou d'un champ. Dans la fonction, le paramètre doit être déclaré du type POINTEUR vers la Structure.

```
void   mettre_date   ( struct date * d )
        { (*d ).a = 2011 ;  (*d ).m = 9 ;(*d ).j = 15 ;
        }
{  struct date d, *e ;
   mettre_date ( &d) ;              e = &d ;
```

3 expressions équivalentes ***d.m , e -> m , (*e).m***

Nouvel opérateur → équivalent au champ de ce qui est pointé par

Pointeurs vers les structures

```
typedef struct date Tdate;
void   mettre_date   ( Tdate * d);
 main ( ) {       Tdate d, *e ;
          mettre_date ( &d) ;
          e = &d;
         }
void   mettre_date   ( Tdate * d )
          {     (*d ).a = 2002 ;
          d →m = 10 ;
          d → j = 9 ;
          }
```

II-9 Exemple ; Déclarations et utilisation

```
int a = 5, *x ;
struct étudiant *z ;          /* z  est de type étudiant */
typedef int *Pe ;             /*  type = pointeur sur entier*/
typedef struct etudiant  Type_etud ;
```

```
{ int a, b, c;
   a = 3
   b = a ;
   b = b + 1 ;
   c = a - b ;
}
```

```
{      int *x, *y, *z ;
       x  = (int*) malloc (int);
       *x = 3 ;
       y = x   ;*y = *y + 1 ;
       z =(int *) malloc(z) ;
       *z = *x - *y ;              }
```

```
void  permut (int *a,
   int *b)
   {   int c =* a;
       *a = *b;     *b = c;
    }
```

```
int nboc(char c, char *ch )
{  int nb = 0, i = 0 ;
   while ( *(ch+i) != '\0' )
        if (*(ch+i) == c) nb ++;
   return nb ;
}
```

```
void  saisie ( float *pt , int n  )
 { int i ;  for ( i = 0; i < n; i++)
            scanf ("%f ", pt + i);
 }
void saisi2 (float *pt, int *pn)
{  int  i=0  ;
   printf («nb de valeurs ?»);
   scanf ( «%d », pn) ;
  for ( i = 0; i < *pn; i + +)
           scanf ("%f ", pt+i );
}
```

```
void  saisie3 (int *pt, int n)
{
  int i  = 0;
  while ( i < n )
    scanf ("%d ", pt + i++);
 }
```

```
#include <stdio.h>
typedef float Tab [ 100 ];
typedef  float * Pr ;
void  saisie3 (Pr pt , int n)
 {  int i  = 0;
    while ( i < n )
      scanf ("%f ", pt + i++ );
 }
void aff (Pr pt, int n)
  {  while ( -- n )
       printf («%f», *(pt+n));
  }
float som( Pr  pt, int n)
  {    float s=0;
       while ( - - n)  s+=*(pt
   +n);
      return s; }
```

```
main ( )
{        int nb ;
         Tab z ;
         printf («entrer nb »);
         scanf («%d» , &nb);
         saisie3 (z, nb) ;
         aff ( z, nb );
      printf («somme = %f» ,
                 som (z, nb) ) ;
}
```