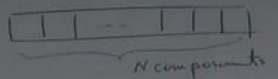
Langage C

Tableaux à un dimensions



déclaration type nom [dimension] ;

soit int T[15].

C reservera 15 * sizeof(int) = 15 * 4 = 60 octet en mémoire

Si A[] (dim n'est pas indiqué l'ordinateur reserve automatiquement).

```
for(i=0;j=0; i<N; i++)
{ T[j] = T[i];
    if (T[i]) ≠ 0   (pour effacer les zéros)
        j++;

}
  N = j  -->  for(
```

Tableaux à 2 dimensions

* déclaration et mémorisation

Type Nom [dim ligne] [dim colonne];

Exemple: (int → long
float → double)

```
int B[10][10];
float B[5][6];        (sizeof(double) = 8)
```

pour B 5 * 6 * 4 = 120 octets (

* Accès aux composants:
* Affichage et Affectation:

```
main(){
    int T[5][4];
    int i,j;          /* Pour l'affectation */
for(i=0; i<5; i++)    /* pour chaque ligne */
  for(j=0; j<4; j++)  /* ... considérer chaque composante */
    printf("Entrez");
    scanf("%d", T[i][j]);

for(i=0; i<5; i++)                /* pour l'affichage */
  for(j=0; j<4; j++)
    printf("%d", T[i][j]);
```

pour $\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}$ → (a b c d e f)

```
for(i=0; i<L; i++)
  {  for(j=0; j<C; j++)
       V[i*C+j] = T[i][j];
  }  dimension de V est L*C
```

Les chaines de caractères

Déclaration

char nomdevariable[longueur];

Ex: char Nom[20];
char phrase[300];

* La représentation interne d'une chaine est terminé par "\0" (NUL). Ainsi pour un texte de n caractere nous devons reserver n+1 octets.
(caractère = octet).

Les C.C constants:

* " Ce texte sera réparti sur 3 lignes."
* " peut être représenté à l'interieur d'une chaine par \".
* ' ⟶ \' peut être " d'une liste de caractère par la séquence d'échappement

{'L', 'i', ... , '\0'}.

* 'x' est un constant, qui a une valeur numérique (1 octet).
* "x" est un tableau de caractère qui contient 2 caract. x et \0 (2 octets)

```
char TXT[ ] = "Hello";
char TXT[ ] = {'H','e','l','l','o','\0'};
```

8.5 Précedence alphabétique et lexicographique

'0' est inf à '=' → on note '0' < '='

↳ Convertir les minuscules dans les majuscules

```
if ( c >= 'a' && c <= 'z')
    c = c - 'a' + 'A';
```

← (Pour une seul lettre)

↳ Ou vice-versa

```
if ( c >= 'A' && c <= 'Z')
    c = c - 'A' + 'a';
```

- Les fonctions de <Stdio.h> printf, scanf, putc, gets

↳ puts(<chaine>) ≡ ecrit la chaine et provoque un retour à la ligne.

```
char T[] = "Bonjour Nsylae";
puts(T);
puts("Bonjour Nsylae");
```

↳ gets(<chaine>) ≡ retour à la ligne est remplacé par '\0'.

```
int Maxi = 1000;
char Ligne[Maxi];
gets(Ligne);
```

/* l'utilisation de scanf est impossible pour lire une phrase contenant un nbr variable de mots */

Les pointeurs

Adressage direct : Accès au contenu d'une variable par le nom de la variable.

Adressage indirect : Accès au contenu d'une variable, en passant par un pointeur qui contient l'adresse de la variable.

↳ Un pointeur : variable spéciale qui peut contenir l'adresse d'une variable

- Si P contient l'adresse d'une variable A, on dit que : "P pointe sur A"
- P peut pointer sur différentes adresses
- Une variable reste fixe lié à la même adresse

'*' ≡ contenu (pour accéder au contenu d'une adresse)

'&' ≡ adresse de &Var fournit l'adresse de Var.

Déclaration d'un POINTEUR :

Type *Nom_Pointeur ; [comme int *P,]

↳ * et & ont la même priorité

le pointeur NUL int *P, P=0;

0 est utilisé pour indiquer qu'un pointeur ne pointe "nulle part"

P1 = P2 ≡ P1 pointe sur le même objet que P2

I - Pointeurs et tableaux

I-1) Adressage des composantes d'un tableau

&tableau[0] est tableau ou &t[0] est t

le nom d'un tableau est un pointeur contenant sur le 1er élément du tableau

int A[10]; int *P; P = A, ou P = &A[0];

P+i pointe sur la ième composante de A (comme P pointe sur A[0])

*(P+1) == A[1] ... *(P+i) == A[i];

P = A ou P++ ✓ mais A = P ou A++ X

A l'adr de A[0]	soit P = A P pointe sur l'élément A[0]
A+i l'adr de A[i]	P+i " A[i]
*(A+i) contenu de A[i]	*(P+i) est le contenu de A[i].

Formalisme tableau

```
main() {
  int T[10] = {4, 5, -7, ..., -5};
  int POS[];
  int i, j;
  for(j=0, j=0; i<10; i++)
  { if( T[i] > 0)
    { Pos[j] = T[i];
      j++;
    }
  }
}
```

Formalisme pointeur

```
main() {
int T[10] = { ... };
int Pos[];
int i, j;
for (j=0; i=0; i<10; i++) {
   if (*(T+i) > 0)
   { *(Pos+j) = *(T+i);
     j++;
   }
}
```

ad et *ad sont des lvalue (modifiables)

Sauf ≡ Puissance + lvalue

t+1 ≡ &T[1] , T+i ≡ &T[i]

T[i] ≡ *(T+i)

tableau de 2 dimen

t[0] ≡ &t[0][]

t[1] ≡ &t[1][]

∄ conversion implicite d'un type pointeur dans un autre

Pointeur Générique — pointeur sans type

Typage des pointeurs peut s'avérer gênant dans certains circonstances comme : fct qui manipule le adresses des objets de type non connu

Void *p ; Void *p

p+1 ; p-q ; p++ ; *p = 10 ; ✗ illégal

On connait pas la taille des objets pointés

Void *p ; int *i ; char *c ;

p = i ✓ ; i = p ; ✗ c = p ✓ ;

Void diag(int *p, int n)

{

int i ;

for(i=0; i<n; i++)

{ *p = 0

p+ = n+1 ;

}

}

Example int T[30][30] ;

diag(T, 30)

Pointeurs sur les fonct°

nom de la fct seul ≡ Son adresse

int (*adr)(double, int) ;

*adr est une fct a 2 arguments (de type double et int) => res = int

int fct(double, int) ;

adr = fct ; OK ✓

Fonct° transmise en argument

I_2) Arithmétique des pointeurs

* Soient P1 et P2 2 pointeurs sur le ^m type.
P1 = P2 ≡ P1 pointe sur le ^m objet que P2.
* Si P pointe sur l'élément A[i] :
✓ P+n pointe sur A[i+n]
✓ P-n pointe sur A[i-n]
✓ P++ ; P pointe sur A[i+1]
✓ P+=n ; P pointe sur A[i+n]
✓ P-- ; " " " A[i-1] (^m pour P-=n;)

int A[10], int *P;
P = A+9 /* dernier element → légal */
P = A+10 /* " " +1 → légal */
P = A+11 /* dernier + 2 */ → illégal
P = A-1 /* premier elmt - 1 */ → illégal

Soient P1 et P2 2 point. qui pointent dans le ^m tableau.
P1 - P2 → nbr de composante comprises entre P1 et P2.
<0 P1 précède P2 ; 0 P1 = P2 ; >0 P2 précède P1
indéfini si P1 et P2 ne pointent pas dans le même tableau.

I_3) Pointeurs et chaines de caractères.

* Utilisons les tableaux de caractères pour déclarer les chaines de caractères que ns voulons modifier
* Utilisés les pointeurs pour le contenue (des chaines cst).

Les structures

But : désigner sous un seul nom un ensemble de valeurs pouvant être de types différents.

Déclaration :

```
Struct   Nom structure
{
        int     x, y, z ;
        float   adr ;
        char    Nom [ ];
} ;
```

✓ Une fois le modèle défini, nous pourrons déclarer des variables du type correspondant `Struct nomstructure Ch ;`

`" Struct nomstruct  Var1, Var2 ; "`

`Ch.x = 7 ;` `ch.Nom[]=" Bonjour " ;`

`printf(" %d ", ch.x) ; scanf(" %d ",&ch.x) ; ch.x ++ ;`

Utilisation globale : on peut affecter à une structure le contenu d'une structure définie à partir du même (nom de modèle) modèle `" Var1 = Var2 ; "`

`Struct nomstructure Vars = { 1,2,3,5, "Nylae" } ;`

On ne peut initialiser notre structure que avec des expressions ctes.

`Typedef int entier ; int n,p ;` <=> `entier n,p ;`

`Typedef int *Ptr ;  int *P1, *P2 ;` <=> `ptr P1, P2 ;`

App aux structures

```
Struct enreg { int num;
               int qte; float prix;
};
```

```
Typedef struct enreg s_enreg;
s_enreg Var1, Var2;
```

```
Struct perso { char Nom[30];
               char Prenom[30];
               float heure[31];
} employe, courant;
```

Réserve les emplacements pour 2 structures nommées employe, courant. Ces derniers comportent 3 champs.

Tableaux de structures :

```
Struct point{
char Nom; int x,y; };
Struct point courbe[50];
```

courbe[i].Nom ; => le nom du pt du i-ème rang du tab courbe

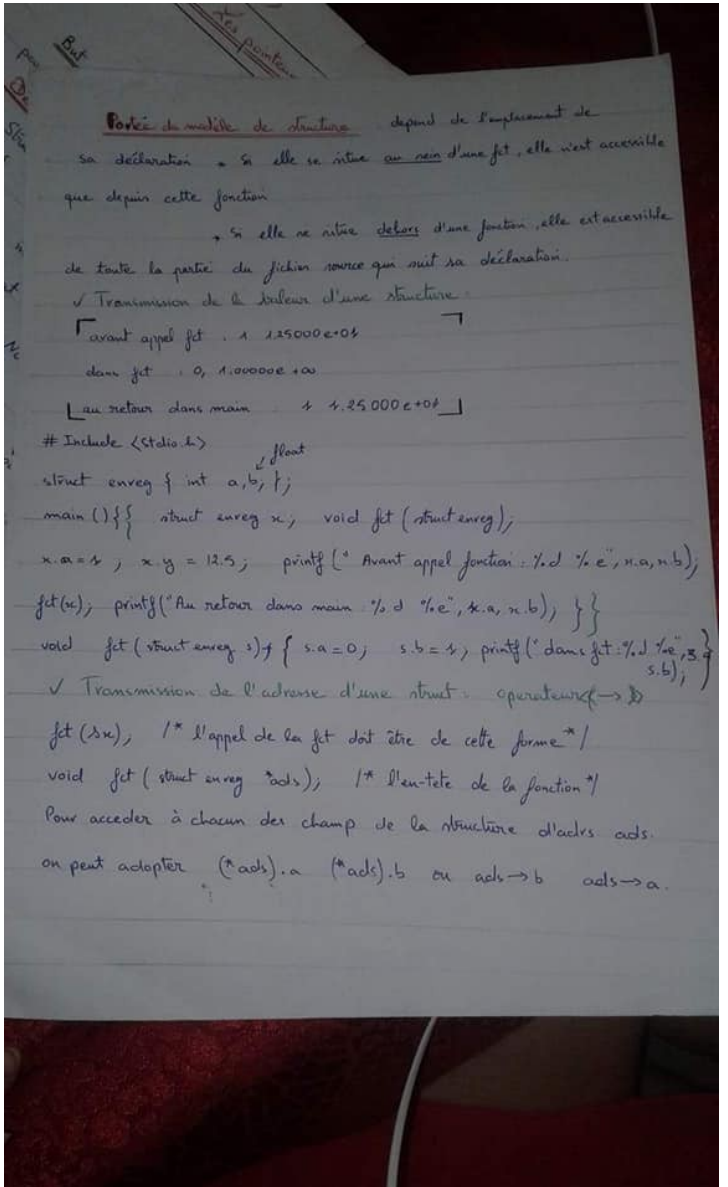
courbe.Nom[i], ✗ FAUX

En d'initialisé° `struct point courbe[50]={ {'A',3,4}, {'C',2,1}};`

Structure comportant autre structure :

```
Struct Date { int jour, mois, annee};
Struct perso { char Nom[] Prenom[]; float heures[31];
               struct date d_embauche, d_poste; } employe, courant;
```

l'appel : employe.d_embauche.jour ;
courant.d_embauche ; (corr à la structure)
courant.d_embauche = employe.d_poste (Affectat° globale).



Portée du nom/tête de structure dépend de l'emplacement de sa déclaration
- Si elle se situe au sein d'une fct, elle n'est accessible que depuis cette fonction
- Si elle se situe dehors d'une fonction, elle est accessible de toute la partie du fichier source qui suit sa déclaration.

✓ Transmission de la valeur d'une structure :

avant appel fct : 1 1.25000e+01

dans fct : 0, 1.00000e+00

au retour dans main : 1 1.25000e+01

```
#include <stdio.h>
struct enreg { int a,b; float }; // float
main (){{ struct enreg x; void fct (struct enreg);
x.a = 1 ; x.y = 12.5; printf (" Avant appel fonction : %d %e", x.a, x.b);
fct(x); printf("Au retour dans main : %d %e", x.a, x.b); }}
void fct (struct enreg s){ s.a = 0; s.b = 1; printf (" dans fct : %d %e", s.a, s.b); }
```

✓ Transmission de l'adresse d'une struct : opérateur &→ *

```
fct (&x); /* l'appel de la fct doit être de cette forme */
void fct ( struct enreg *adrs); /* l'en-tete de la fonction */
```

Pour accéder à chacun des champ de la structure d'adrs adrs.

on peut adopter (*adrs).a (*adrs).b ou adrs→b adrs→a.

```
#include <stdio.h>
struct enreg { int a; float b; };
main(){
    struct enreg x;
    Void fct (struct enreg *ads);
    x.a = 1;   x.b = 12.5;
    printf("\n avant appel fct : %d %e", x.a, x.b);
    fct(&x);
    printf("\n au retour dans main : %d %e", x.a, x.b);
}
Void fct( struct enreg *ads)
{ ads->a = 0; ads->b = 1;
  printf("\n dans fct:%d %e", ads->a, ads->b);
}
```

Avant appel fct : 1 1.250000e+01
dans fct 0 1.000000e+00
Au retour dans main :
0 1.000000e+00

Transmission d'une structure en valeur de retour d'une fct

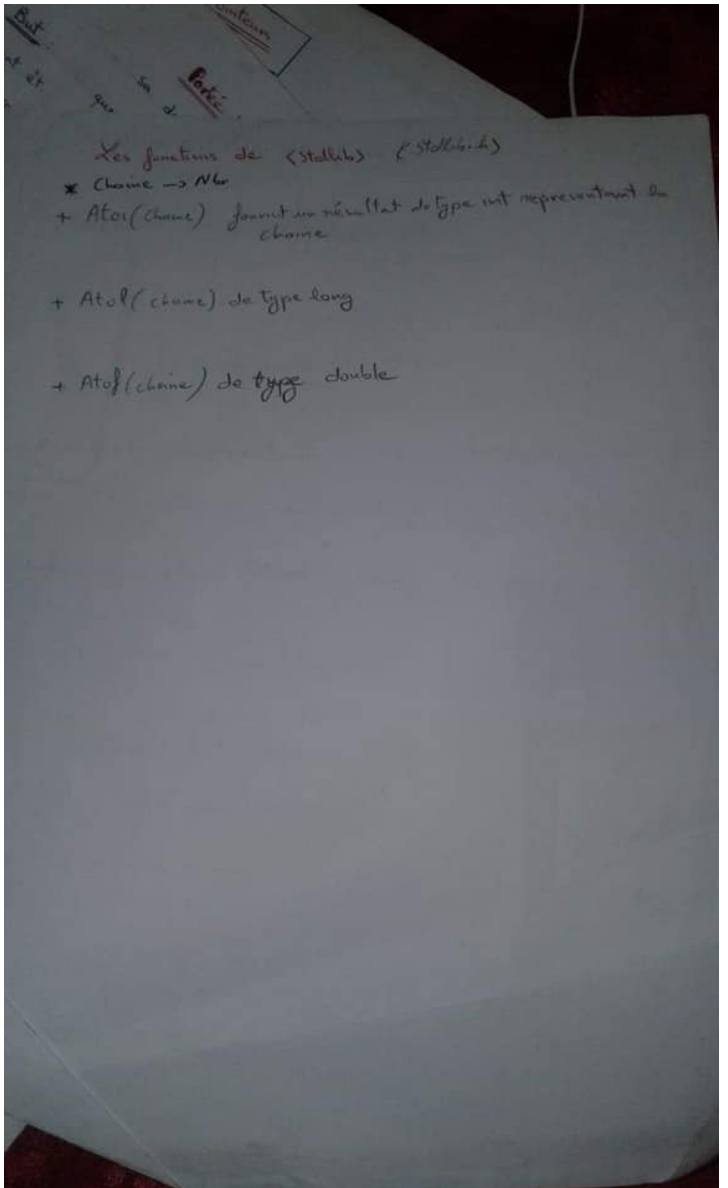
```
Struct enreg fct (...)
{
    struct enreg s;

    return s;
}
```

Les fonctions de string => pour le traitement de chaines de caractères.

+ strlen(<s>) longueur de la chaine sans compter '\0' final { for(i=0; s[i]; i++) i = longueur }
+ strcpy(<s>,<t>) copie <t> vers <s>
+ strcat(<s>,<t>) Ajouter <t> à la fin de <s>
+ strcmp(<s>,<t>) Compare <s> et <t> Pour les chaînes
+ strncpy(<s>,<t>,n) copie n caractere de <t> -> <s> n = longueur minimal de <t>
+ strncat(<s>,<t>,n) Ajoutes n caract. de <t> à <s>

pour strcomp(<s>,<t>) donne un résultat negatif si <s> precede <t>
-> 0 si <s> egal <t>
-> positif si <s> suit <t> -> <t> > <s>

+ strchr (chaine, caractere) cherche dans la chaine la premiere position où apparait le caractere.
+ strrchr(C, S) même chose que (strchr) mais à partir de la fin.
+ strstr(C, S_chaine) recherche la premiere occurrence complete de la S-chaine


Les fonctions de (Stdlib) (Stdlib.h)
★ Chaine → Nb
+ Atoi(chaine) fournit un résultat de type int représentant la chaine
+ Atol(chaine) de type long
+ Atof(chaine) de type double