Exercice

```
Type structure personne
nom:chaine;
numero:chaîne;
Fin structure;

Type structure carnet
T[200]: personne;
nb:entier;
Fin structure ;

procedure saisir(d/r P: personne)
debut
Afficher("le nom SVP ! ");
Entrer(p.nom) ;
afficher("le num SVP ! ");
Entrer(P.numero) ;
Fin procedure
{Nous verrons lors de la prochaine
séance la traduction exacte en C des
param d/r En classe nous avons réalisé
un passage en d}
```

```
Procedure affiche(p:personne)
Debut
afficher("le numéro de telephone de ", P.nom," est ",
p.numero);
Fin;

Algorithme annuaire
C1: carnet;
i,n: entier;
Debut
c1.nb<-0;
repeter
afficher("combien de personnes allez vous saisir?");
Entrer(n);
jusqu'à (n>0 et n<200);
pour(i de 0 a n-1) Faire
saisir(C1.T[i]) ;
C1.nb<-C1.nb+1;
pour(i de 0 a n-1) Faire
affiche(c1.T[i]);
Fin;
```

Exercice

```
#define TN 33
#define TNUM 17
#define TAILLE 200
typedef struct {
         char nom [TN];
         char numero[TNUM];
} personne;
typedef struct
{        personne T[TAILLE];
         int nb;
} carnet;
void saisir(personne *p)
{
      printf("le nom SVP ! ");
       scanf("%s",p->nom)  ;//ou
      scanf("%s",  (*p).nom);
       printf("le num SVP ! ");
       scanf("%s",p->numero)  ;
}
```

```
void affiche(personne p)
{ printf("le numéro de telephone de %s est %s\n",p.nom, p.numero);}
void main()
{        carnet c1;
         int i,n;
         c1.nb=0;
        do{
            printf("combien de personnes allez vous saisir?");
            scanf("%d",&n);
          } while (n<=0 || n>=200);
         for(i=0; i<n;i++)
         {
                 saisir(&c1.T[i]) ;
                 c1.nb++;
         }
         for(i=0; i<c1.nb;i++)
                 affiche(c1.T[i]);
}
```

Si on allait accéder à une variable int dans un scanf comme age par exp:
scanf("%d",&(*p).age) ; //ou
scanf("%d",&p->age) ;

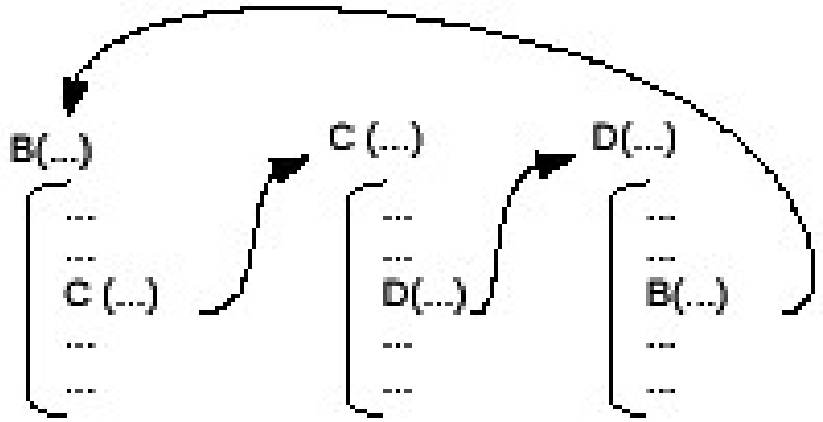
Chapitre 3: La récursivité

AÏCHA EL GOLLI
aicha.elgolli@essai.ucar.tn

Définition

Une procédure ou une fonction est dite récursive si elle fait appel à elle même, directement ou indirectement.

Récursivité directe



Récursivité indirecte

Définition

- Appel récursif --> Boucle
- Pour éviter les boucles infinies, les appels récursifs doivent être conditionnels (à l'intérieur d'un SI ou un SINON ou un TQ ...etc).
- **Remarque :** Il est rare qu'un programmeur doive écrire lui-même une fonction récursive. Cependant, il est profitable d'être capable d'écrire des fonctions récursives simples et de les simuler pour comprendre le principe.

Exemple1 : La fonction factorielle

n! = n * (n-1) * (n-2) * .... * 1 pour n > 0 et 0! = 1

n! = n * (n-1)! pour n > 0 et n! = 1 pour n = 0

```
Fonction Fact( n:entier ) : entier
Début
   SI (n=0) alors
     retourne(1) ;        /* cas particulier */
   SINON
     retourne(n * Fact( n-1 )) ; /* cas général */
   FSI
Fin
```

```
#include <stdio.h>
long Fact (int n)
{
  if(n==0) return 1;
  else
        return(n* Fact(n-1));
}
```

Déroulement pour n=4 :

1- 4! = 4 * 3!

2- 3! = 3 * 2!

3- 2! = 2 * 1!

4- 1! = 1 * 0!

5- 0! = 1 (cas particulier)

l'exécution i attend la terminaison de l'exécution i+1 pour continuer son traitement

Exemple 2

Déterminer du k ième chiffre à partir de la droite d'un entier n > 0 :

Le 3ième chiffre à partir de la droite de 8724 est 7
Le 5ième chiffre à partir de la droite de 21327 est 2
Le 6ième chiffre à partir de la droite de 21327 est 0

Écrire une fonction récursive chiffre(n, k) qui permet de retourner le k-ième chiffre à partir de la droite de n.

Observations :

1. Si k = 1, la solution est triviale. Le chiffre recherché est
   le dernier chiffre de n, c'est-à-dire n mod10 : si (k = 1) retourne n mod 10 ; C'est la condition d'arrêt.
2. Si k > 1, exemple k = 3, on a :
   chiffre (8724, 3)
   <==> chiffre (872, 2) (872 est 8724 / 10, 2 est 3-1)
   <==> chiffre (87, 1) (87 est 872 / 10, 1 est 2-1)

Exemple 2

```
Fonction chiffre( n:entier,k:entier ) : entier
Début
   SI (k=1) alors
     retourne(n mod 10) ; /* cas particulier */
   SINON
     retourne(chiffre( n div 10, k-1 )) ; /* cas général */
   FSI
Fin
```

```
#include <stdio.h>
int chiffre ( int n, int k )
{
        if ( k == 1 ) /* le premier chiffre à partir de la droite */
            return n % 10 ;
         else
            return chiffre ( n / 10, k - 1 );
        }
}
```

Exemple 3: recherche d'un élément dans un tableau

Version itérative: écrire une fonction recherche qui prend un entier à chercher, un tableau d'entier et sa taille et retourne vrai si l'entier existe faux sinon

```
Fonction recherche(t []:entier, n : entier, val : entier) : booléen
variables
trouvé : booléen ;
cpt : entier ;
début
cpt ← 0 ;
trouvé←FAUX ;
tant que (non trouvé et cpt < nbr) faire
        trouvé←  (tab[cpt] = val) ;
        cpt ← cpt +1 ;
ftq
retourne(trouvé) ;
fin ;
```

Idée d'une solution récursive

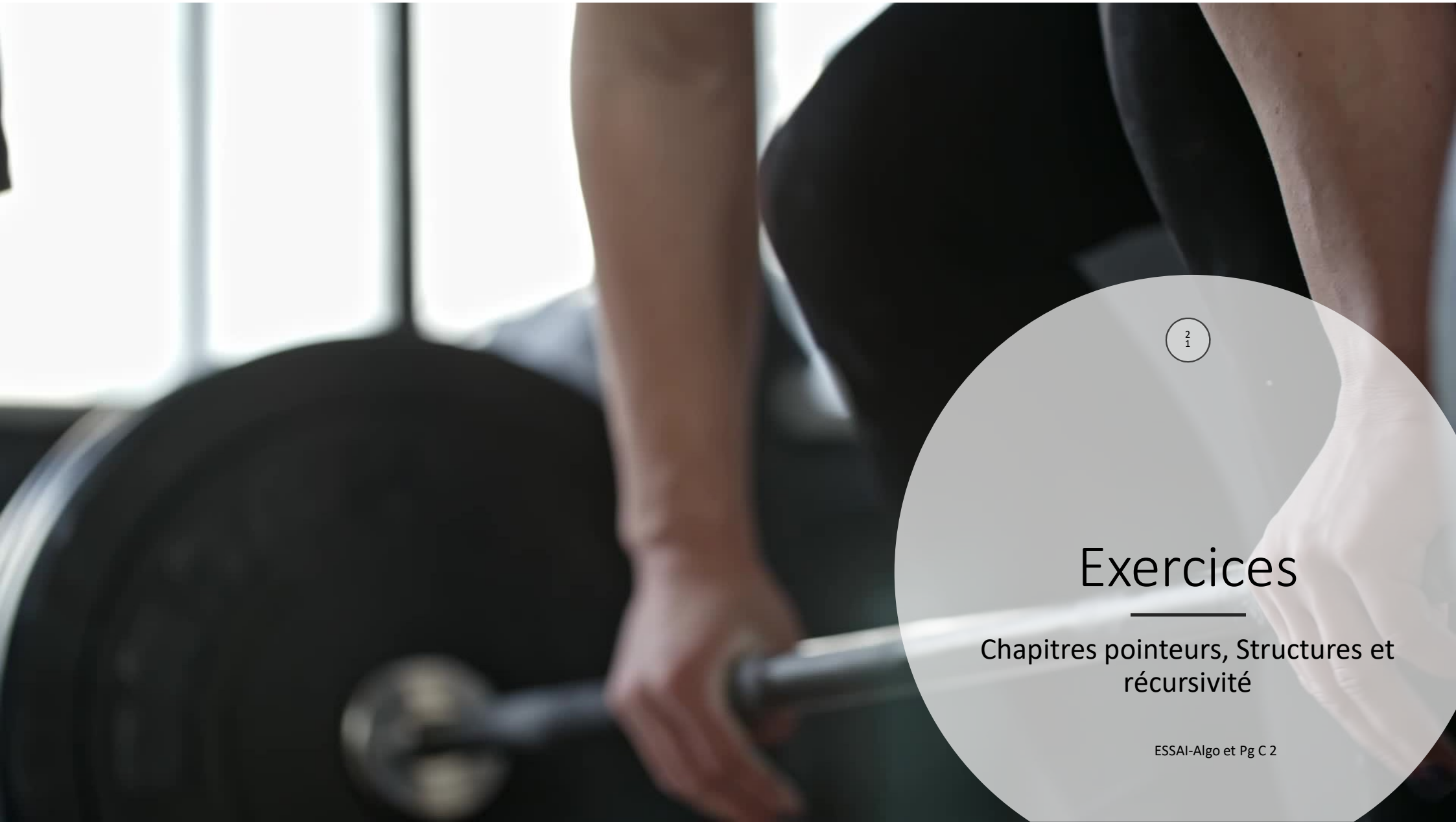
- **recherche(t, n, val)**
  - est-ce que val est la dernière valeur du tableau?
  - si oui, fin (retourne Vrai), sinon, reprendre dans le tableau sans la dernière valeur :

    recherche(t, n-1, val)

    → appel récursif, possible pour n > 0
- **n= 0** condition d'arrêt

```
Fonction recherche(t []:entier, n : entier, val : entier) : booléen
Début
Si(t[n-1]=val) alors retourne vrai;
Si(n=1) retourne faux;
Retourne( recherche(t,n-1,val));
fin ;
```

```
int recherche(int t[], int n, int val)
{
  if(t[n-1]==val)      return 1;
  if(n==1)             return 0;
   return(recherche(t,n-1,val));
}
```



Exercices

Chapitres pointeurs, Structures et récursivité

Exemple de problème avec pointeurs

```
void saisie(int *a, int *b){
printf("donner la valeur de x= ");
scanf ("%d",a);
printf("donner la valeur de y= ");
scanf ("%d",b);
}
void permut(int *x,int *y){
int c;
c=*x;
*x=*y;
*y=c;}
void affichage(int *x, int *y){
printf("la valeur de x=%d la valeur de y= %d", *x, *y);}
void main()
{
    int *x,*y;
    saisie(x,y);
    permut(x,y);
    affichage(x,y);
}
```

Exercice

On souhaite écrire un algorithme de gestion des étudiants très simplifié :

1. Créer une structure nommée Etudiant pouvant contenir ces informations :

nom, prenom, date de naissance et téléphone. Les nom et prénom peuvent contenir 32 caractères, le téléphone peut contenir 13 caractères. La date de naissance est une structure composée du jour, le mois et l'année de naissance (des entiers).

2. Créer une nouvelle structure qui va représenter une classe. Cette structure nommée Classe contiendra un tableau T de 35 étudiants et un compteur dim indiquant le nombre d'étudiants dans le tableau.

3. Ecrire une procédure creEtud qui permet de saisir un étudiant passé en argument.

4. Ecrire une procédure aff qui affiche les informations contenues dans la structure Etudiant passée en argument,

5. Ecrire une fonction récursive nbEtud qui étant une classe donnée et une année de naissance renvoie le nombre d'étudiants nés durant cette année.

6. En utilisant toutes les fonctions et procédure définies précédemment écrire un programme principale qui demande de saisir n étudiants, qui les ajoute dans une classe puis qui affiche son contenu. Cet algorithme affichera à la fin le nombre d'étudiants nés en une année n saisie par l'utilisateur.

Pour cela on doit tout d'abord déclarer une variable de type classe, une autre variable de type Etudiant.