

**ALGORITMIQUE
AVANCE LANGAGE C**

TCSRIT LICENCE 1



KONAN HYACINTHE



CHAPITRE 3 : LA RECURSIVITE

3.1 DEFINITION

Une famille d'objet est dite récursive, si dans sa définition il est fait référence à la famille elle-même.

Pour un langage de programmation, nous dirons qu'il autorise la récursivité si un sous-programme peut s'appeler lui-même directement ou indirectement à travers un autre sousprogramme.

Pour les langages à structure de bloc, le problème de la conservation des contextes successifs est résolu grâce à la **pile d'exécution dynamique** : les variables locales et les paramètres sont empilés à chaque appel récursif du sous-programme.

3.2 RECURSIVITE DIRECT / RECURSIVITE INDIRECT

Réversivité directe	Réversivité indirecte ou croisée	
Procedure P ; DEBUT P ; FIN;	Procedure A ; DEBUT C ; FIN;	Procedure B ; DEBUT A ; FIN;
	Procedure C ; DEBUT B ; FIN;	

Notons que dans le cas de la réversivité croisée, il existe un problème syntaxique de déclaration d'une procédure avant l'autre :

Procedure A ; DEBUT B ; FIN;	Procedure B ; DEBUT A ; FIN;
--	--

La directive **forward** sert à résoudre ce problème. Lors de la déclaration, cette directive sert à **déclarer syntaxiquement l'en-tête** d'une procédure qui sera **déclarée en totalité plus loin**. Cette directive permet d'utiliser la réversivité croisée en particulier :

Procedure B ; forward ; Procedure A ; DEBUT B ; FIN;	Procedure B ; DEBUT A ; FIN;
--	--

3.3 EXEMPLES DE PROBLEMES RESOLUS RECURSIVEMENT

Exemple 1: Calcul de la factorielle

- $\text{fact}(3) = 3 \times 2 \times 1$
- $\text{fact}(N) = N \times \text{fact}(N-1)$ la fonction fait appel à elle-même vrai pour $N > 0$
- $\text{fact}(0) = 1$ condition d'arrêt

Fonction **fact(N)** retourne (entier)

paramètre (D) N : entier

variable résultat : entier

début

si (N = 0) **alors**

retourne (1)

sinon

résultat $\leftarrow$ (N $\times$ **fact(N-1)**)

retourne (résultat)

fsi

fin

Exemple 2: le PGCD de deux entiers

- $\text{pgcd}(A,B) = \text{pgcd}(B, A \bmod B)$ appel récursif, vrai pour $B > 0$
- $\text{pgcd}(A,0) = A$ condition d'arrêt

Fonction **pgcd(A,B)** retourne (entier)

paramètres (D) A,B : entiers

variable résultat : entier

début

si B = 0 {test d'arrêt de la récursion} **alors**

retourner (A)

sinon

résultat $\leftarrow$ **pgcd(B, A mod B)** {appel récursif}

retourne (résultat)

fsi

fin

Exemple 3: recherche d'un élément dans un tableau

Version itérative

Fonction **recherche(val, tab, nbr)** retourne (booléen)

paramètres (D) val, nbr : entiers

(D) tab : **tableau** [1, MAX] d'entiers

variables trouvé : booléen ; cpt : entier

début

cpt $\leftarrow$ 0 ; trouvé $\leftarrow$ FAUX

tant que (non trouvé et cpt < nbr) **faire**

cpt $\leftarrow$ cpt +1

trouvé $\leftarrow$ (tab[cpt] = val)

ftq

retourne(trouvé)

fin

Idée d'une solution récursive

- **recherche(val, tab, nbVal)**
 - est-ce que val est la dernière valeur du tableau?
 - si oui, fin (retourne Vrai), sinon, reprFINre dans le tableau sans la dernière valeur :
recherche(val, tab, nbVal - 1)
 → appel récursif, possible pour nbVal > 0
- **nbVal = 0** condition d'arrêt

Fonction recherche(val, tab, nbVal) retourne (booléen)

paramètres (D) val : entier

(D) tab : **tableau** [1, MAX] d'entiers

(D) nbVal : **entier**

début

si (nbVal = 0) {test d'arrêt de la récursion} **alors**

retourne (FAUX)

sinon

si (tab[nbVal] = val) **alors**

retourne (VRAI)

sinon

retourne (**recherche(val, tab, nbVal - 1)**) {appel récursif}

fsi

fsi

fin

simulation

unTab

1	5	8	2
---	---	---	---

recherche(5, unTab, 4)

1er appel :

1. val1=5, tab1=unTab, nbr1=4

Récapitulation

- Fonction récursive ?
 - Fonction qui s'appelle elle-même.
- Comment éviter les appels infinis ?
 - Trouver le test qui va déterminer l'arrêt.
- Comment les écrire?
 - Définir le cas général : il contiFINra l'appel récursif.
 - Définir le cas particulier : c'est le test d'arrêt.

version récursive alternative

→ Comparer les enchaînements récursifs (très inefficace)

Fonction recherche(val, tab, nbr) retourne (booléen)

paramètres (D) val : entier
 (D) tab : tableau [1, MAX] d'entiers
 (D) nbr : entier

début

```

si (nbr = 0) alors
    retourne (FAUX)
sinon si (recherche( val, tab, nbr - 1)) alors
    retourne (VRAI)
sinon
    retourne (tab[nbr] = val)
fsi
fsi
  
```

fin

Exemple 4: inversion d'une chaîne de caractères

Exemple : abcde

1. Enlever le premier caractère	a
2. Inverser le reste de la chaîne → appel récursif	edcb
3. Rajouter le premier en fin de chaîne	edcba

Condition d'arrêt ?

Inverser une chaîne de longueur 1 = ne rien faire

Exemple 4: simulation

1er appel : inverser "algo"

1. enlever "a"
2. inverser "lgo"
3. rajouter "a"

Procédure d'inversion

Procédure inverser (mot, nbcar)

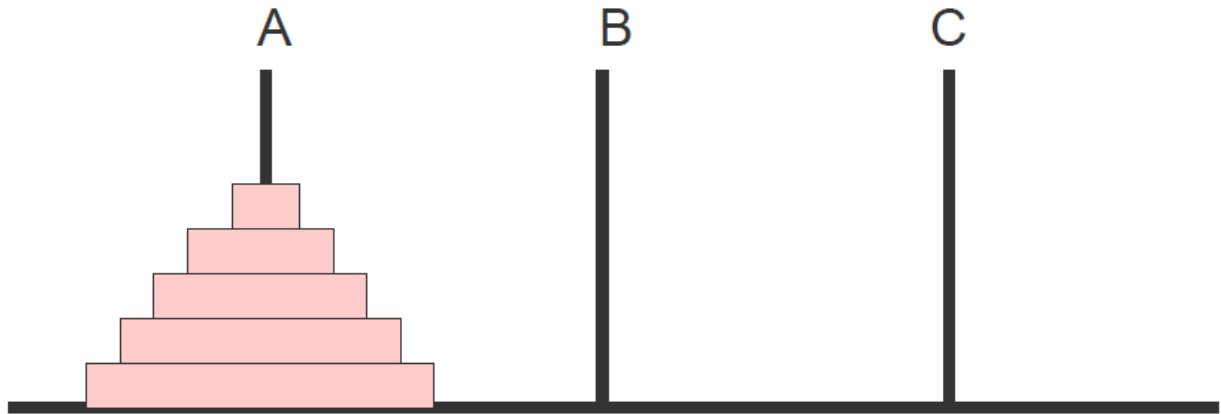
paramètres (D/R) mot : tableau [1, MAX] de caractères
 (D) nbcar : entier

Exemple 5: les tours de hanoi

Problème :

transférer les N disques de l'axe A à l'axe C, en utilisant B, de sorte que jamais un disque ne repose sur un disque de plus petit diamètre.

Ici : $N = 5$



Idée d'une solution récursive

- supposons que l'on sache résoudre le problème pour $(N-1)$ disques :
 1. on déplace $(N-1)$ disques de A vers B en utilisant C
 2. on fait passer le N -ième grand disque en C {déplacer le plus grand}
 3. on déplace les $(N-1)$ disques de B vers C en utilisant A
- c'est l'idée générale de la récursion :
 - par exemple, si je sais calculer $\text{fact}(N-1)$, je sais calculer $\text{fact}(N)$
{ « réduction » du problème, jusqu'à la condition d'arrêt }
- pour les tours de hanoi, le problème de taille N donne naissance à deux problèmes de taille $(N-1)$

Squelette d'une procédure récursive

```
1. Procédure hanoi (source, but, aux, nb)
   paramètres
début
   2. si nb = 1 alors
      déplacer un disque de source vers but
   sinon
      3. hanoi (source, aux, but, nb - 1)
      4. déplacer un disque de source vers but
      5. hanoi (aux, but, source, nb - 1)
   fsi
fin
```

Simulation de hanoi(A,C,B,3)

[il manque les flèches d'appel et de retour!]

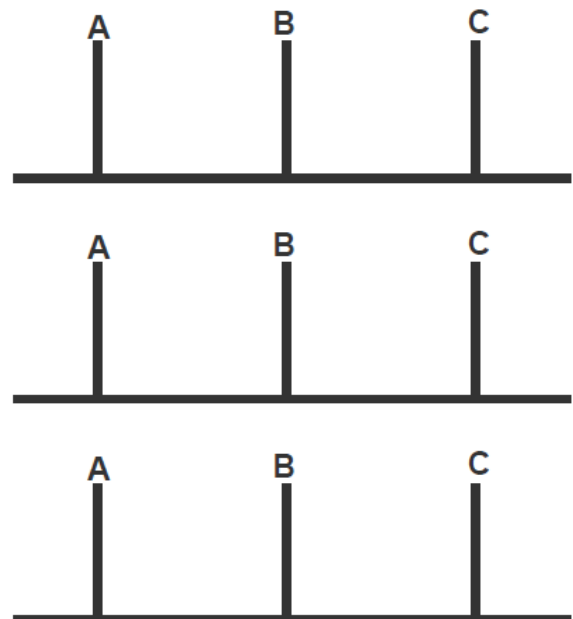
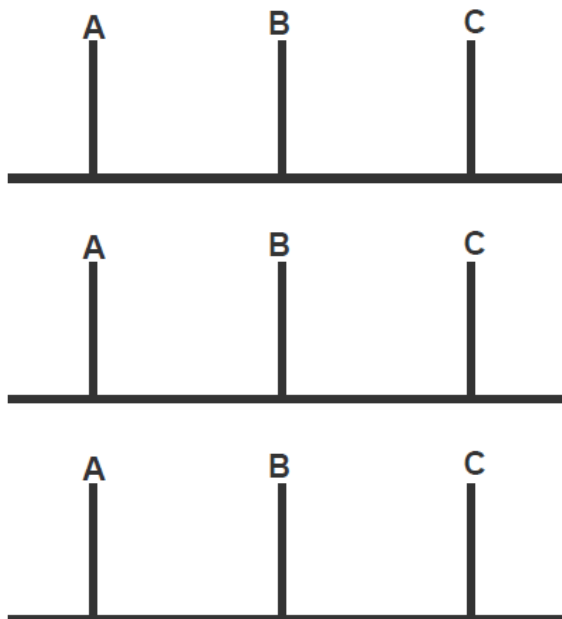
1^{er} appel : hanoi(A,C,B,3)

1. Faux

2^{ème} appel : hanoi(A,B,C,2)

3^{ème} appel : hanoi(A,B,C,1) 1. vrai 2. déplacer 1 disque de A à C	2^{ème} hanoi(A,B,C,2) 4. déplacer 1 disque de A à C 5. hanoi(B,C,A, 2)	1. faux 3. hanoi(A,C,B,1) 4. déplacer 1 disque de A à B 5. hanoi(C,B,A,1)
4^{ème} appel : hanoi(C,B,A,1) 1. vrai 2. déplacer 1 disque de C à B	5^{ème} appel : hanoi(B,C,A, 2) 1. faux 3. hanoi(B,A,C,1) 4. déplacer 1 disque de B à C 5. hanoi(A,C,B,1)	6^{ème} appel : hanoi(B,A,C,1) 1. vrai 2. déplacer 1 disque de B à A
		7^{ème} appel : hanoi(A,C,B,1) 1. vrai 2. déplacer 1 disque de A à C

Simulation de hanoi(A,C,B,3)



Récapitulation

- Fonction récursive ?
 - Fonction qui s'appelle elle-même.
- Comment éviter les appels infinis ?
 - Trouver le test qui va déterminer l'arrêt.
- Comment les écrire ?
 - Définir le cas général : il contiendra l'appel récursif.
 - Définir le cas particulier : c'est le test d'arrêt.

Un dernier exemple

Soit un tableau de pixels à deux dimensions $N \times N$. On représente les pixels allumés à l'aide de la valeur 1 et les pixels éteints à l'aide de la valeur 0. Ecrire une fonction récursive `surfaceRégion` qui prend en paramètres (entre autres) les coordonnées x et y d'un pixel et qui renvoie la surface de la région à laquelle appartient ce pixel. La surface est évaluée ici en terme de nombre de pixels allumés. Une région est un ensemble de pixels allumés liés par continuité horizontale, verticale ou diagonale.

Exemple, ce tableau ci-contient 3 régions et `surfaceRégion` doit retourner

- 5 pour le point (3,4),
- 2 pour le point (1,2)
- 0 pour le point (5,5)
- 4 pour le point (5,1)

		x				
		1	2	3	4	5
y	1	1	0	1	1	1
	2	1	0	0	1	0
	3	0	0	0	0	0
	4	0	1	1	0	0
	5	1	0	1	1	0