بسم الله الرحمن الرحيم

اسأل الله ان اجدكم في احسن حال و الصحة, انتم و ذويكم

السلام عليكم و رحمه الله و بركاته

Salam, j’ espère que vous allez bien ainsi que votre famille

La récursivité

Rappel et exercices

- ✓ Principe
- ✓ Implémentation
- ✓ Exemple
- ✓ Finalité
- ✓ Illustration d'une solution récursive
- ✓ Exercices 1-2-3 série 2
- ➢ développement des solutions des exercices

développement des solutions des exercices

EX1 : Ecrire un sous-programme récursif qui calcule la somme $U_n = 1+2^4+3^4+\ldots+n^4$.

Exemple illustratif : n = 4

$U_4 = 1 + (2x2x2x2) + (3x3x3x3) + (4x4x4x4)$

$U_4 = 1+ 16 + 81 + 256$

$U_4 = 354$

Dans la solution itérative, nous suivons la formule donnée dans l'exercice

Solution itérative

```
Int somiterative (int n)
{ int som = 0;
      i ;
      for (i = 1; i <= n; i++)  som = som + (i * i * i * i);
  return som;
}
```

Solution récursive de Ex 1

Reprenons la formule donnée dans l'exercice , alors :

$$U_n = 1+2^4+3^4+\ldots+n^4$$

- Nous pouvons observer que U_n est composée de la somme de U_{n-1} et de n^4

- De la, nous pouvons redéfinir Un en :

$U_n = U_{n-1} + n^4$

tel que $U_{n-1} = U_{n-2} +(n-1)^4$ et donc il faut développer celle de U_{n-2}

-

-

$U_1 = 1$

$U_0 = 0$ c'est le cas trivial (cas particulier ou il n y est plus nécessaire que la fonction s'appelle à elle-même et qui permet d' arrêter la récursivité).

Solution récursive de Ex 1

Exemple illustratif : n = 4 Pour calculer U_4 on utilise la formule précédente

$U_4 = U_3 + (4 * 4 * 4 * 4)$ mais faut il avoir la valeur de U_3 ,donc il faut maintenant calculer U_3

$U_3 = U_2 + (3 * 3 * 3 * 3)$ donc il faut calculer U_2

$U_2 = U_1 + (2 * 2 * 2 * 2)$ donc il faut calculer U_1

$U_1 = U_0 + (1 * 1 * 1 * 1)$ donc il faut calculer U_0

$U_0 = 0$ c'est le cas trivial (le cas sans appel à la fonction)

Ce n'est pas fini, en réalité, à chaque appel, on arrête momentanément la fonction appelante pour exécuter la fonction appelée

C'est-à-dire :

- Pour calculer U_4 , il faut calculer U_3 ⟶ donc on arrête U_4 et on l'a met dans une *pile pour y revenir plus tard

- Et ainsi de suite jusqu'à rencontré U_0 dont on peut l' exécuter sans l' arrêter

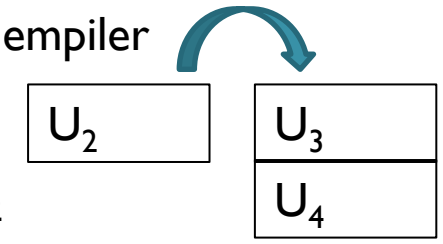
- maintenant il faut revenir vers les fonctions précédentes et remonter à U_4

Solution récursive de Ex 1

Exemple illustratif : n = 4 Pour calculer U_4 on utilise la formule précédente

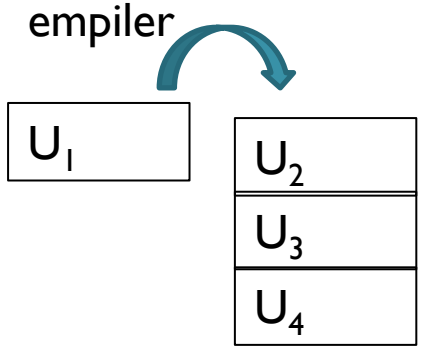
$U_4 = U_3 + (4 * 4 * 4 * 4)$ mais faut il avoir la valeur de U_3 on arrête l' exécution de U_4 , et on l'empile et on appel U_3

$U_3 = U_2 + (3 * 3 * 3 * 3)$ on arrête l' exécution de U_3 et on l'empile et on appel U_2



$U_2 = U_1 + (2 * 2 * 2 * 2)$ on arrête l' exécution de U_2 et on l'empile et on appel U_1

$U_1 = U_0 + (1 * 1 * 1 * 1)$ on arrête l' exécution de U_1 et on l'empile et on appel U_0



$U_0 = 0$ c'est le cas trivial (sans récursivité, la fonction peut se terminer et retourner son résultat)

Maintenant, il faut revenir vers les appels précédents, en dépilant de la pile , une à une

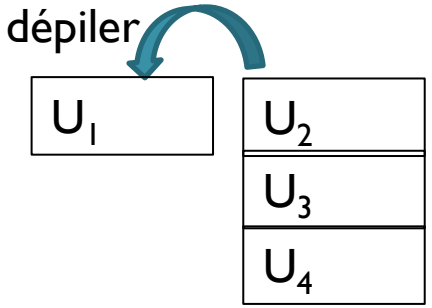
Solution récursive de Ex 1

Maintenant, il faut revenir vers les appels précédents, en dépilant de la pile , une à une

$U_0 = 0$

$U_1 = U_0 + (1 * 1 * 1 * 1)$, on remplace la valeur de U_0 on termine U_1

$U_1 = 1$

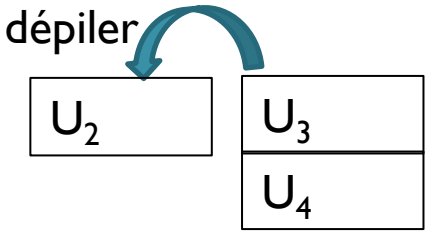


$U_0 = 0$

$U_1 = 1$

$U_2 = U_1 + (2 * 2 * 2 * 2)$, on remplace la valeur de U_1 on termine U_2

$U_2 = 1 + 16 = 17$



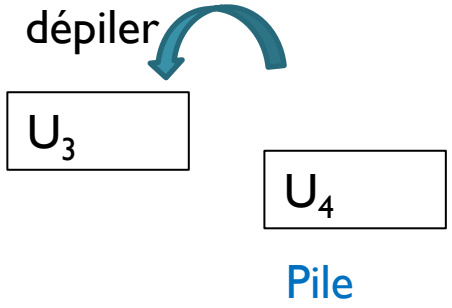
$U_0 = 0$

$U_1 = 1$

$U_2 = 17$

$U_3 = U_2 + (3 * 3 * 3 * 3)$, on remplace la valeur De U_2 et on termine U_3

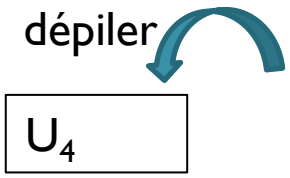
$U_3 = 17 + 81 = 98$



Solution récursive de Ex 1

Maintenant, il faut revenir vers les appels précédents, en dépilant de la pile , une à une

$U_0 = 0$
$U_1 = 1$
$U_2 = 17$
$U_3 = 98$
$U_4 = U_3 + (4 * 4 * 4 * 4)$, on remplace la valeur
De U_3 et on termine U_4
$U_4 = 98 + 256 = 354$



Pile vide

La Pile est vide, donc nous avons calculer la fonction appelante qui est U_4

*Remarque

- Une Pile n'est rien d'autre qu'un espace mémoire, avec une gestion particulière, on met les éléments un à un à partir du sommet de la pile et quand on retire un élément, on le retire aussi du sommet

- Ce qu'on appelle « first in last out », premier entrer dernier sortie.

Solution récursive de Ex 1

Maintenant qu'on a illustré le fonctionnement de cette fonction, procédant à sa définition

Solution récursive

```
int somrecursive (int n)
{
   if (n <= 0)
      return 0;
   else
      return (n*n*n*n)+ somrecursive (n-1);
}
```

✓Vous avez remarqué que la fonction travaille sur des entiers donc, c'est évident que le résultat est définie en tant que tel et que la fonction retourne un entier .

✓ Nous avons des calculs, donc nous avons défini la solution autant qu'une fonction.

✓Le cas trivial, défini par la condition (n<= 0), englobe le n = 0 et aussi les cas impossibles avec n < 0