Université Mohammed V
Faculté des Sciences
Rabat

Algorithmique II

-TD 2- Récursivité

Exercice 1

Écrire une fonction qui permet de retourner la liste des diviseurs d'un entier passé en paramètre.

```
Fonction diviseurs(N : entier) : { Procédure diviseurs(N : entier) est également possible }
Variables
        i : entier
début
        écrire("Les diviseurs de ", N, "sont : ")
        pour i ← 1 à N faire
                si N mod i = 0 alors
                        écrire( i )
                fsi
        fpour
fin
```

Exercice 2

Écrire une fonction qui permet de calculer la factorielle d'un entier positif passé en paramètre :

1- Version itérative
2- Version récursive

1- Version Itérative

```
Fonction fact(N : entier) : entier
Variables
        F, i : entier
début
        F ← 1
        pour  i ← 2 à N faire
                F ← F * i
        fpour
        Retourner ( F )
fin
```

2- Version Récursive

```
Fonction fact(N : entier) : entier
début
        si N > 0 alors
                Retourner ( N * fact(N-1) )
        fsi
        Retourner ( 1 )
fin
```

ou bien

Université Mohammed V
Faculté des Sciences
Rabat

```
Fonction fact(N : entier) : entier
Début
        si N = 0 alors
                Retourner ( 1 )
        sinon
                Retourner ( N * fact(N-1) )
        fsi

Fin
```

Exercice 3

Écrire une fonction récursive qui permet de déterminer le nombre de combinaisons possibles de p éléments parmi n, C_n^p. On note les formules suivantes :

$$\begin{cases} C_n^n = 1 \\ C_n^1 = n \\ C_n^p = C_{n-1}^p + C_{n-1}^{p-1} \end{cases}$$

```
Fonction comb(n : entier, p : entier) : entier
début
        si p = n alors
                Retourner ( 1 )
        fsi
        si p = 1 alors
                Retourner ( n )
        fsi
        Retourne ( comb(n-1, p)+comb(n-1, p-1) )
fin
```

Exercice 4

En utilisant une approche récursive :

1- Écrire un algorithme qui permet d'afficher les éléments d'un tableau.

```
Algorithme affichage_Tableau
    Fonction display(T : Tableau réel, N : entier) :
    début
        si N > 0 alors
                display( T, N-1 ) {on commence par l'appel récursif avant écrire pour afficher les
                écrire( T(N-1) )                                         éléments dans l'ordre}
        fsi
    fin

Variables
        Taille : entier ← 4
        Tab : Tableau réel ← {14, 3, -8, 12.5}
début
        écrire("Les éléments du tableau sont : ")
        display(Tab, taille)
```

```
fin
```

2- Écrire une fonction qui additionne les éléments du même tableau.

```
Fonction add_tab(T : Tableau réel, N : entier) : réel {N > 0}
début
   si N = 1 alors
            Retourner ( T(0)  )
   fsi
   Retourner ( T(N-1) + add_tab(T, N-1) )
fin
```

3- Écrire une fonction qui calcule le max (le min) dans le même tableau

Première solution avec fonction max

```
Fonction _max(a : réel, b : réel) : réel
début
   si a < b alors
            Retourner ( b )
   sinon
            Retourner ( a )
   fsi
fin
Fonction max_tab(T : Tableau réel, N : entier) : réel {N > 0}
début
   si N = 1 alors
            Retourner ( T(0) )
   fsi
   Retourner (  _max( T(N-1) + max_tab(T, N-1) )   )
fin
```

Deuxième solution sans fonction max

```
Fonction max_tab(T : Tableau réel, N : entier) : réel {N > 0}
Variables
   max : réel
début
   si N = 1 alors
            Retourner ( T(0) )
   fsi

   sinon
            max ← max_tab(T, N-1)
            si max < T(N-1) alors
                     max ← T(N-1)
            fsi
            Retourner ( max )
   fsi
fin
```

Fonction min

```
Fonction min_tab(T : Tableau réel, N : entier) : réel
Variables
   min : réel
```

```
début
    si N = 1 alors
            Retourner ( T(0) )
    fsi

    sinon
            min ← min_tab(T, N-1)
            si min > T(N-1) alors
                    min ← T(N-1)
            fsi
            Retourner( min )
    fsi
fin
```

Exercice 5

On considère la fonction récursive suivante :

Fonction fonc1 (a : entier, b : entier) : entier
début
si b = 0 alors
Retourner (a)
sinon
Retourner (** *fonc1(b, a mod b)* ***)
fsi
fin

1- Calculer fonc1(5, 3), fonc1(12, 9), fonc1(7, 14)

fonc1(5, 3) = fonc1(3, 2) = fonc1(2, 1) = fonc1(1, 0) = 1

fonc1(12, 9) = fonc1(9, 3) = fonc1(3, 0) = 3

fonc1(7, 14) = fonc1(14, 7) = fonc1(7, 0) = 7

2- Que permet de calculer cette fonction ?

Le pgcd (plus grand commun diviseur)

Exercice 6

Écrire une fonction récursive qui retourne la somme des chiffres d'un entier N donné. Exemple : (123 == > 1 + 2 + 3 = 6)

```
Fonction sum_digits(N : entier) : entier
début
    si N < 10 alors
            Retourner ( N )
    sinon
            Retourner ( (N mod 10) + sum_digits(N div 10) )
    fsi
fin
```

Exercice 7

Écrire un algorithme récursif permettant de convertir un entier en système binaire. (2=> 10 et 7 => 111)

```
Procédure toBinary(N : entier)
Début
        si N = 0 alors
                écrire(0)
        sinon si N = 1 alors
                écrire(1)
        sinon
                toBinary(N div 2)
                écrire(N mod 2)
        fsi
Fin
```

NB. Le test (N=1) est facultatif car il permet seulement d'afficher 1 (toBinary(1) => 1) au lieu de 01 (=>01) lorsque N=1.
L'appel récursif (toBinary(N div 2)) doit apparaitre avant (écrire(N mod 2)) pour avoir un résultat correct. Il faut noter que cette récursivité n'est pas terminale.

Université Mohammed V
Faculté des Sciences
Rabat

Exercice 8

On considère la fonction récursive suivante :

Fonction fonc2 (a : entier, b : entier) : entier
début
  si a = 0 alors
    Retourner (** b **)
  fsi
  Retourner (** *fonc2(a - 1, a + b)* **)
fin

1- Calculer fonc2(3,5), fonc2(12,0), fonc2(-7, 14)

fonc2(3,5) = fonc2(2,8) = fonc2(1,10) = fonc2(0,11) = 11

fonc2(12,0) = fonc2(11,12) = fonc2(10,23) = fonc2(9,33) = fonc2(8,42) = fonc2(7,50) = fonc2(6,57) = fonc2(5,63) = fonc2(4,68) = fonc2(3,72) = fonc2(2,75) = fonc2(1,77) = fonc2(0,78) = 78

fonc2(-7,14) = fonc2(-8,7) = fonc2(-9,-1) =….. c'est une boucle infinie

2- Corriger cette fonction et proposer une fonction itérative équivalente

Correction

Fonction fonc2 (a : entier, b : entier) : entier
début
  si a < 0 alors
    Retourner (-1) ***{La valeur -1 est arbitraire, il faut retourner un entier}***
  fsi
  si a = 0 alors
    Retourner (** b **)
  fsi
  Retourner (** *fonc2(a - 1, a + b)* **)
fin

Version itérative 1

Fonction fonc2 (a : entier, b : entier) : entier
début
  si a < 0 alors
    Retourner (-1)
  fsi
  tant que a > 0 alors
    b ← a+b
    a ← a-1
  ftq
  Retourner (** b **)
fin

Université Mohammed V
Faculté des Sciences
Rabat

Version itérative 2

Fonction fonc2 (a : entier, b : entier) : entier
début
si a < 0 alors
Retourner (-1)
fsi
***Retourner (** *a*(a+1)/2 + b* **)**
fin

Exercice 9

On considère la fonction récursive suivante :

Fonction fonc3 (a : entier) : entier
début
si a > 0 et a < 5 alors
fonc3(a + 1)
fonc3(a - 4)
fsi
fin

si la fonction fonc3(1) est appelée dans un programme, combien de fois la fonction **fonc3** sera-t-elle appelée avant que le programme passe aux instructions suivantes.

Cette fonction est à récursivité double (multiple)
Les valeurs possibles de a sont : 1, 2, 3 et 4 donc si a est différent de ces 4 valeurs on fera aucun appel récursif. Pour schématiser la récursivité double (où il y a deux appels récursif), on utilise les arbres binaires.

- fonc3(1)
  - fonc3(2)
    - fonc3(3)
      - fonc3(4)
        - fonc3(5)
        - fonc3(0)
      - fonc3(-1)
    - fonc3(-2)
  - fonc3(-3)

J'ai stérilisé en rouge les appels de fonction qui ne fait plus d'appel récursif càd quant a<1 ou a>4
Maintenant pour savoir le nombre d'appel de la fonction **fonc3**, il suffit de compter le nombre des nœuds dans l'arbre binaires qui vaut **9 appels**.

Exercice 10

On considère les deux fonctions suivantes :

Fonction fun1 (N : entier) : entier
début
si N <= 0 alors
retourner *1*
sinon
retourner *fun2(N - 1)*
fsi
fin

Fonction fun2 (N : entier) : entier
début
si N <= 0 alors
retourner *2*
sinon
retourner *fun1(N - 2)*
fsi
fin

1- Que peut-on dire de ces deux fonctions ?

On peut remarquer que chacune de ces deux fonctions ne fait appelle à elle-même, cela veut dire qu'aucune des deux n'est récursive. En revanche, on peut remarquer qu'en combinant les deux fonctions, on obtient un comportement récursif. Car chaque fonction fait appel à l'autre fonction. On appelle cela : récursivité mutuelle et les deux fonctions sont mutuellement récursives.

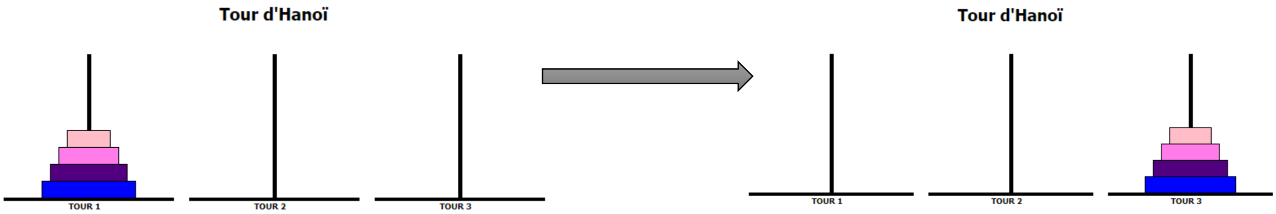
2- Déterminer de fun1(10) et fun2(10)

fun1(10) = fun2(9) = fun1(7) = fun2(6) = fun1(4) = fun2(3) = fun1(1) = fun2(0) = 2

fun2(10) = fun1(8) = fun2(7) = fun1(5) = fun2(4) = fun1(2) = fun2(1) = fun1(-1) = 1

Exercice 11

Écrire une fonction récursive double « hanoi » qui prend en argument le nombre de pièces à déplacer, le nom du piquet 1 (depart), le nom du piquet 2 (destination) et le nom du piquet 3 (intermediaire). Ladite fonction doit permettre d'afficher les instructions pour déplacer les n pièces en respectant les règles du jeu de Hanoi.



Le jeu de Hanoi (de récursivité double) est un exemple concret qui montre la force des fonctions récursives. Générer un algorithme déterminant toutes les étapes (mouvements) nécessaires pour déplacer les disques tout en respectant les règles de Hanoi est vraiment très compliqué si on adopte une approche itérative.

Ici on donne seulement la solution, par contre les règles du jeu ainsi que l'idée d'appliquer la récursivité est largement documenter sur internet (jouer en ligne, vidéos démonstratives … sont disponibles).

Il faut noter que l'algorithme suivant ne donne pas que la solution pour réussir le jeu mais la solution optimale avec le minimum de mouvements possibles.

```
Fonction hanoi(n: entier, A: Piquet, B: Piquet, C: Piquet) {Déplacer n disques du piquet A à piquet C }
Début
    si n > 0 alors
        hanoi(n-1, A, C, B)  {Déplacer n-1 disques du piquet A à piquet B }
        écrire("Déplacer le disque ", n, " de ", A, " à ", C")
        hanoi(n-1, B, A, C)  {Déplacer n-1 disques du piquet B à  piquet C }
    fsi
Fin
```