Travaux Pratiques d'algorithmique n°3

Cours d'Informatique de Deuxième Année

—Deug SSM M3-MI—

Récursivité

L'objet de cette séance est d'introduire la notion de récursivité. Dans un premier temps, on s'intéressera à plusieurs exemples basiques qui permettent de se familiariser avec cette technique. Ensuite, on étudiera des problèmes plus élaborés.

▸ **Exercice 1.** *Ecrire une fonction récursive qui affiche les N premiers entiers dans l'ordre croissant. Idem dans l'ordre décroissant.*
Que constatez-vous ?

▸ **Exercice 2.** **(Somme et produit récursifs)** *Ecrire deux fonctions qui calculent, de façon récursive, la somme et le produit de deux nombres entiers* `a` *et* `b` *positifs ou nuls.*
Indication : pour calculer la somme, nous disposons des relations suivantes :

$$\begin{cases} SOMME(a,0) = a \ , \\ SOMME(a,b) = SOMME(a,b-1)+1 \ . \end{cases}$$

Evaluer les complexités des algorithmes programmés.

▸ **Exercice 3.** **(Coefficients du binôme)** *On sait que les coefficients du binôme C_n^p définis par :*

$$\begin{cases} C_n^p = \frac{n!}{p!\,(n-p)!} & si\ n \geq p \ , \\ C_n^p = 0 & si\ n < p \ , \\ C_n^p = 1 & si\ p = 0 \ . \end{cases}$$

vérifient la récurrence suivante :

$$C_n^p = C_{n-1}^{p-1} + C_{n-1}^p \qquad pour\ 0 < p < n \ .$$

Ecrire une fonction récursive qui calcule C_n^p .

▸ **Exercice 4.** **(Polynômes de Legendre)** *Ecrire une fonction récursive permettant de calculer les polynômes de Legendre définis par :*

$$\begin{cases} P_0(x) = 1 \ , \\ P_1(x) = x \ , \\ P_n(x) = \dfrac{(2\,n-1)\,x\,P_{n-1}(x) - (n-1)\,P_{n-2}(x)}{n} \ . \end{cases}$$

▸ **Exercice 5.** *Ecrire une fonction* `Puissance` *en appliquant la définition suivante :*

$$b^0 = 1 \ , \qquad b^n = b\,b^{n-1} \ .$$

Ecrire une fonction `PuissanceBis` *qui tient compte des relations suivantes :*

$$\begin{cases} b^n = (b^{n/2})^2 & \text{si } n \text{ est pair} \ , \\ b^n = b\,(b^{(n-1)/2})^2 & \text{si } n \text{ est impair} \ , \\ b^n = 1 & \text{si } n = 0 \ . \end{cases}$$

Evaluer la complexité dans les deux cas.

▸ **Exercice 6.** **Inversion d'une chaine récursivement** *Le but de cet exercice est d'implémenter une fonction de prototype* `Chaine InversionChaine(Chaine ch)` *permettant d'inverser une chaîne de caractères. Pour cela, nous utiliserons une nouvelle fonction de prototype* `void _InversionChaine(Chaine ch, Chaine ch_inverse)` *qui sera récursive. Elle sera appelée par* `InversionChaine`*. En effet, la première fonction aura pour but d'allouer l'espace mémoire afin de stocker la chaîne inversée. Le pointeur sur la zone mémoire réservée sera passé en deuxième argument à la fonction récursive afin qu'elle puisse y écrire la chaîne inversée. Il ne restera plus qu'à renvoyer le résultat. Exemple :*

"bonjour" nous donnera "ruojnob".

▸ **Exercice 7.** *Programmer de façon récursive, les fonctions suivantes :*

- `int strcmp(Chaine cs, Chaine ct)` *qui compare la chaîne* `cs` *à la chaîne* `ct`*; On retournera une valeur négative si* `cs` $<$ `ct`*, nulle si* `cs` $=$ `ct`*, et positive si* `cs` $>$ `ct`.
- `char * strchr(Chaine cs, Chaine c)`*; On retournera un pointeur sur la première occurence de* `c` *dans* `cs`*, ou NULL si* `c` *ne figure pas dans* `cs`.
- `char * strrchr(Chaine cs, Chaine c)`*; On retournera un pointeur sur la dernière occurence de* `c` *dans* `cs`*, ou NULL si* `c` *ne figure pas dans* `cs`.