Remarques sur le premier contrôle de TD :

Démêlons les confusions en tous genres ...

Licence 1 MASS semestre 2, 2006/2007

La première remarque est qu'il faut apprendre le cours, faire les exercices et poser des questions lorsque vous ne comprenez pas ...

1 Entrées-Sorties d'un algorithme

Pour calculer la moyenne des notes de contrôle et le nombre de note au dessus de la moyenne, nous avons fait les petits algorithmes suivants ... Tout d'abord la moyenne :

Algorithme Moyenne():
variable n, i : entier; x, m : réel
début
  écrire("Donner nombre de notes : ")
  lire(n)
  $m \leftarrow 0$
  pour i **de** 1 **à** n **faire**
    écrire("Donner note : ")
    lire(x)
    $m \leftarrow m + x$
  fin pour
  si $n \neq 0$ **alors**
    $m \leftarrow \frac{m}{n}$
  fin si
  écrire("La moyenne est : ", m)
fin

Vous noterez qu'avant de faire la division on vérifie que n n'est pas nul ...

Maintenant un petit algorithme pour calculer combien de notes sont au dessus de la moyenne. L'idéal serait d'utiliser l'algorithme Moyenne. Le premier problème qu'on rencontre est que moyenne ne retourne rien, par contre, il affiche la moyenne une fois calculée, donc si on veut utiliser Moyenne pour le second algorithme il faut que l'algorithme commence comme suit :

Algorithme Supérieur_Moyenne():
variable m : réel
début
  Moyenne()
  écrire("Donner moyenne : ")
  lire(m)

... suite ...
fin

Ceci manque réelement d'élégance est n'est vraiment pas pratique du tout. En effet, l'execution de cet algorithme commence par executer Moyenne, puis l'utilisateur doit lire la moyenne à l'écran de la machine, la noter sur un bout de papier quelconque. Ensuite l'algorithme demande de donner cette valeur qu'il faut saisir au clavier ...

C'est quand même plus simple de faire retourner la moyenne, c'est-à-dire un réel, à l'algorithme Moyenne et de le récupèrer directement dans l'algorithme Supérieur_Moyenne de la façon suivante :

Algorithme Moyenne():réel
variable n, i : entier; x, m : réel
début
  écrire("Nombre de notes : "**)**
  lire(n**)**
  $m \leftarrow 0$
  pour i **de** 1 **à** n **faire**
    écrire("Donner note : "**)**
    lire(x**)**
    $m \leftarrow m + x$
  fin pour
  si $n \neq 0$ **alors**
    $m \leftarrow \frac{m}{n}$
  fin si
  retourner m
fin

Algorithme Supérieur_Moyenne():
variable m : réel
début
  $m \leftarrow$ Moyenne()
  ... suite ...
fin

Vous constaterez bien que ce qui est retourné est bien un réel et en aucun cas ("La moyenne est : ", m) qui est une chaîne de caractères ... Normalement la différence entre **retourner** et **écrire** devrait être plus claire maintenant ...

Il faut maintenant terminer l'algorithme Supérieur_Moyenne. On a saisi les notes une fois pour calculer la moyenne mais c'etait dans un autre algorithme, on n'a pas d'autre moyen que de les redemander à l'utilisateur :

Algorithme Supérieur_Moyenne():entier
variable m, x : réel; n, i, c : entier
début
  $m \leftarrow$ Moyenne()
  $c \leftarrow 0$
  écrire("Nombre de notes : "**)**
  lire(n**)**
  pour i **de** 1 **à** n **faire**
    écrire("Donner note : "**)**
    lire(x**)**
    si $x \geq m$ **alors**
      $c \leftarrow c + 1$
    fin si

fin pour
retourner c
fin

On évite de faire la même erreur et d'afficher c au lieu de le retourner, en effet on pourrait avoir besoin de cette valeur dans un autre algorithme. De toutes façons, si on veut vraiment l'afficher, on écrit un autre algorithme tout simple :

Algorithme Afficher_Supérieur_Moyenne():
début
écrire("La moyenne est : ", Moyenne(), " et le nombre de notes au dessus de cette moyenne est : ", Supérieur_Moyenne()**)**
fin

Maintenant réfléchissons à ce qui se passe quand on execute l'algorithme Supérieur_Moyenne ... On execute Moyenne(), il va falloir saisir les notes une par une, ensuite on redemande les notes une par une pour compter combien de notes sont au dessus de la moyenne de la classe ... donc on passe plus de temps à saisir des notes qu'à faire le calcul à la main ... On va donc modifier tout ça pour n'avoir à saisir les notes qu'une seule fois. La solution est d'utiliser un nouvel algorithme pour stocker les notes dans un tableau et de donner ce tableau en paramètre de Moyenne et de Supérieur_Moyenne :

Algorithme Entrer_Notes():tableau de réel
variable T : tableau de réel; n, i : entier
début
écrire("Nombre de notes : "**)**
lire(n**)**
pour i **de** 0 **à** $n-1$ **faire**
écrire("Donner note : "**)**
lire($T[i]$**)**
fin pour
retourner T
fin

On doit aussi modifier les deux autres algorithmes :

Algorithme Moyenne(t : tableau de réel; n : entier):réel
variable i : entier; m : réel
début
$m \leftarrow 0$
pour i **de** 0 **à** $n-1$ **faire**
$m \leftarrow m + t[i]$
fin pour
si $n \neq 0$ **alors**
$m \leftarrow \frac{m}{n}$
fin si
retourner m

fin

Algorithme Supérieur_Moyenne(t : tableau de réel; n : entier):entier
variable m : réel; c : entier
début
  $m \leftarrow$ Moyenne()
  $c \leftarrow 0$
  pour i **de** 0 **à** $n-1$ **faire**
    si $t[i] \geq m$ **alors**
      $c \leftarrow c+1$
    fin si
  fin pour
  retourner c
fin

Et maintenant si on veut afficher tout nos calculs, on n'a plus quà faire appel à un algorithme tout simple. On suppose ici qu'un algorithme **taille** prend un tableau en paramètre et retourne sa taille.

Algorithme Afficher_Calcul():
variable T : tableau de réels
début
  $T \leftarrow$ Entrer_notes()
  écrire("La moyenne des notes est : ", Moyenne(T, taille(T)),"et le nombre de notes au dessus de cette moyenne est : ", Supérieur_Moyenne(T, taille(T))**)**
fin

Au final, on n'a rentré les notes qu'une seule fois et on a pourtant réussit à faire tout ce qu'on voulait ! Notez que mettre le tableau de notes en paramètre de l'algorithme Afficher_Calcul serait une abhération étant donné que c'est précisement le rôle de cet algorithme de les faire saisir au clavier par l'utilisateur.

Pour les entrées/sorties d'un algorithme, on a le choix entre lire/paramètres, et écrire/retourner. L'exemple précédent montre que ces méthodes sont très différentes et ne sont pas adaptées aux mêmes cas. Dans l'idée de pouvoir réutiliser un algorithme dans plusieurs situations différentes (comme calculer une moyenne ou compter les notes au dessus de la moyenne de la classe) il vaut mieux toujours éviter les lire et écrire et privilégier les paramètres et les retourner. Au pire, si on tient vraiment aux lire et écrire, on peut toujours faire un autre algorithme qui lit les entrées, les donne en paramètre à notre algorithme général qui retourne les résultats pour l'autre algorithme puisse les écrire ...

Dernière remarque sur les sorties : le type de ce que retourne l'algorithme doit impérativement être donné en en-tête de l'algorithme (après les ':') et respecté au moment du **retourner** !!! Il faut aussi bien s'assurer que quelle que soit les données de l'algorithme il retournera quelque chose de cohérent : par exemple dans Moyenne, on retourne bien toujours un réel même si $n = 0$ ... on a prévu le cas ...

2 Indentation et séparation de code

Notons au passage que dans cet exemple on a appliqué le principe de séparation de code, c'est-à-dire qu'on a décomposer tout le travail depuis saisir les notes jusqu'à afficher la moyenne et le nombre de notes qui lui sont supérieures en petites parties :

- Afficher_Calcul : saisi les notes, appel Moyenne et Supérieur_Moyenne, afficher le résultat
- Moyenne : calcul la moyenne,
- Supérieur_Moyenne : appel Moyenne : ne refait pas le calcul lui même, compte le nombre de notes supérieur à la moyenne des notes,

Chacun de ces algorithmes fait relativement peu de choses, par conséquent ils sont simples à écrire et à vérifier, et risquent moins d'être erronnés. De plus on peut réutiliser les algorithmes principaux : Moyenne et Supérieur_Moyenne dans d'autres contextes.

Contrairement à ce que certains croient, la séparation de code n'a rien à voir avec les passages à la ligne ou l'indentation ... on pourrait réecrire tous les algorithmes précédents sur une seule ligne et sans espaces, on aurait quand même fait de la séparation de code. Par contre ça serait illisible.

L'indentation et les passages à la ligne servent uniquement à rendre les algorithmes lisibles, en particulier quand on a des **si** imbriqués, on aligne verticalement le **si**, le **sinon** et le **finsi** correspondant en décalant vers la droite les instructions à effectuer si la condition est vraie (ou fausse dans le **sinon**). Un algorithme lisible est toujours plus simple à vérifier et à comprendre !

3 Paramètres et variables

Comme on l'a vu précédement, les paramètres sont donnés (entre parenthèses en en-tête) à un algorithme par l'extérieur, c'est à dire par un autre algorithme qui l'appel. Les paramètres sont les données à partir desquelles l'algorithme doit travailler pour produire un résultat. Ils ne doivent pas être confondus avec les variables de l'algorithme (déclarées sur la ligne **variable :**) qui correspondent plutôt à des outils utilisés pour produire le résultat. Par exemple, pour faire un gateau, on a besoin d'ingrédients : les paramètres et on va utiliser des récipients pour placer ces ingrédients, les mélanger etc. Ces récipients correspondent aux variables : on en a besoin pendant l'execution de la recette (algorithme), mais une fois le gateau terminé, on n'en a plus besoin et on n'en parle plus. Les variables n'existent qu'en le **début** et le **fin** d'un algorithme. Il arrive toutefois qu'une variable créee soit retournée, comme par exemple quand un algorithme crée un tableau. Dans ce cas c'est un peu comme pour la recette de la mousse au chocolat, on a besoin de retourner le plat dans lequel on a mit la mousse sinon on se heurte à quelque ennuieux problèmes de transport ... c'est pareil avec les tableaux. Attention, si un tableau crée comme variable d'un algorithme peut

exister en dehors de cet algorithme c'est uniquement parcequ'il est **retourné** à la fin ...

4 Les finsi ne sont pas uniquement décoratifs

Algorithme Test(a : réel):réel
début
  si $a \geq 0$ **alors**
    si $a \leq 10$ **alors**
      $a \leftarrow 1$
    sinon
      $a \leftarrow 0$
    fin si
  fin si
  retourner a
fin

Algorithme TestBIS(a : réel):réel
début
  si $a \geq 0$ **alors**
    si $a \leq 10$ **alors**
      $a \leftarrow 1$
    fin si
  sinon
    $a \leftarrow 0$
  fin si
  retourner a
fin

Les algorithmes **Test** et **TestBIS** ne diffèrent que d'un **finsi** qui n'est pas placé au même endroit dans l'un et dans l'autre. Le tableau suivant donne, suivant les valeurs du paramètre a, les valeurs retournées par ces algorithmes.

a	$]-\infty, 0[$	$[0, 10]$	$]10, +\infty[$
Test	a	1	0
TestBIS	0	1	a

Vous pouvez donc constater que ces deux algorithmes sont très différents et que la place des **finsi** est *extrèmement* importante. Ils ne sont pas là pour décorer mais bien parcequ'ils sont nécessaires à la bonne compréhension de l'algorithme. Les oublier ou les mettre n'importe où (agglutinés à la fin par exemple ...) rend l'algorithme *in-com-pré-hen-si-ble* !!!

D'autre part, lorsque les instructions à réaliser au cours d'une itération (de boucle **pour** ou **tant que**) contiennent un **si** il est *absolument impératif* que le **finsi** qui le termine soit placé *avant* l'indicateur de fin de boucle (**finpour** ou **fintantque**). Dans le cas contraire, l'algorithme n'a tout simplement aucun sens.

Il faut donc être très précis avec les **fin...** en général car ils sont indispensables et mal placés ils rendent incorrect même le meilleur des algorithmes.