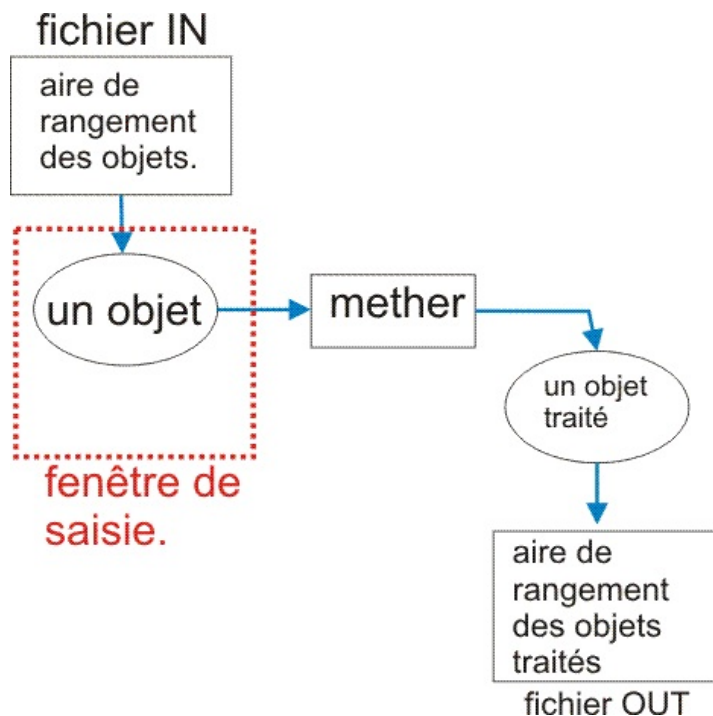


## De l'Ordre Informatique.

L'ordre Quintilien n'a d'autre objectif que de doter l'humain d'une intelligence globale et planétaire. Elle doit être capable d'anticiper les effets de tout phénomène en phase d'évolution, ou de constitution. Elle doit doter l'humanité, faite d'un corps unique que j'ai appelé Diquupankù, de son cerveau. Il est utile d'analyser, comment cet objectif peut ne jamais être atteint.

Dans ce cas, cela libère une obligation de destruction d'humanité, qui sera suivie tôt ou tard d'une nouvelle construction, aux mêmes objectifs.

Reprenons le schéma simplifié de tâche : [stache.gif](#)



Premier constat : inutile de penser à traiter, si l'on n'a pas réuni les matériaux à traiter, si l'on n'a pas vérifié que ces matériaux sont compatibles avec le traitement.

L'aire de rangement IN, OUT est fondamentale, comme les racines de tout ce qui se développe et vit.

La **fenêtre** est ce qui permet de saisir un objet nouveau, puis le ranger dans un fichier OUT, qui deviendra le fichier IN d'un traitement.

Mais une **fenêtre** sert aussi à redéfinir les attributs d'un objet, dont le modèle est pris dans un fichier IN.

Elle sert encore, à vérifier si un objet est compatible avec la méthode, le traitement qui lui sera appliqué. Enfin toute méthode, tout traitement est le comment d'une tâche de quintilien. Tous sont inclus dans le **mether** universel.

Tout cela conduit à un aspect important, pour tout langage qui se veut formateur de l'intelligence :

1- Toute intelligence quintilienne, sait déduire la chaîne des transformations, obligatoirement subies par l'objet à traiter, pour donner en sortie l'objet résultat. Pour cela, une vision binaire, en hexadécimal est nécessaire pour explorer tout objet en entrée et en sortie. Car toute transformation, ne porte que sur position et valeur des bits.

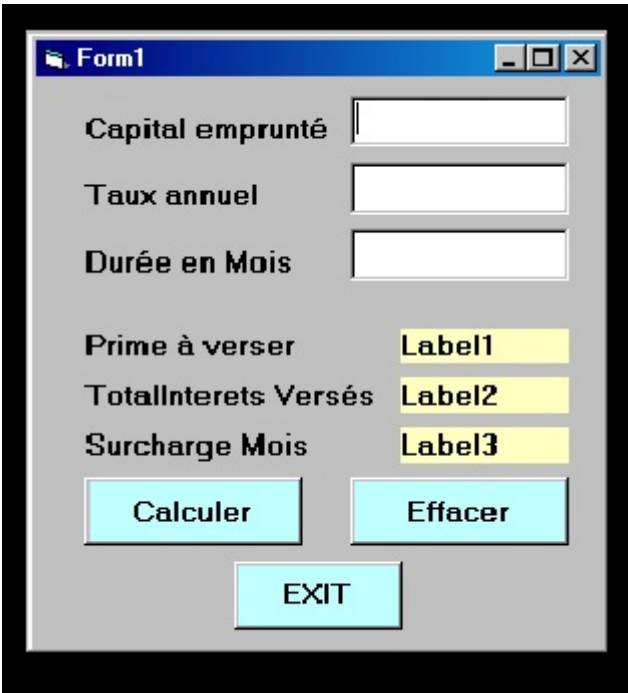
2- Les ordres de lecture et d'écriture sur fichiers doivent être simples et intuitifs. Directs, sans préparation d'aucune sorte : Les ordres classiques **Read**, **Write** suffisent. La documentation sur le langage, doit fournir des exemples qui tournent et que l'on peut reproduire, ou exploiter. Il faut noter que tous les ordres Read ou Write, sont formés à partir des interruptions du BIOS. C'est à ce niveau

qu'exactitude et performance doivent porter.

3- Pour toute partie de programme, un traitement, doit s'écrire avec le langage classique de programmation. Une variation de ce langage, ne doit jamais remettre en cause ce qui fut fait, mais fixer les erreurs des fonctions existantes, ajouter des fonctions éprouvées, comme autant de nouvelles instructions du langage. L'orthographe doit rester invariable, la syntaxe également, la sémantique doit se préciser, s'affiner...

4- On constate, que deux types de fenêtres couvrent tous les besoins : celles qui sont, explicites, connues dans le langage **HTML** :

```
<form ... action="traitement" ...>  
...  
</form>
```



Celle qui sont, implicites, connues dans les langages Visual. Vous disposez d'une forme pleine sur laquelle s'affichent les dessins des formes particulières. Comme vu ci-contre en Visual Basic 6.

Certaines formes ont des programmes liés qui permettent de choisir dans une liste : par exemple la forme **Drive**, la forme **Dir**, la forme **File**. Il faut penser à des fonctions écrites en Assembleur ou C, qui sont associées à des icônes. On ne voit que les icônes.

D'autres n'ont aucun programme, on ne donne que l'icône, il faut écrire, développer

l'**action** qui devra traiter ce que reçoit la **forme**.

Ce que l'on attendait d'une programmation en **VISUAL**, c'est qu'elle reste une programmation classique, exploitant le langage support classique, ici Basic. C'est qu'elle permette d'associer à tout programme, à toute fonction, à tout sous-programme, une icône (forme de base fournie : carré, rectangle, cercle, ovale) que le programmeur pourrait compléter. Enfin les routines toutes prêtes, incluses dans une librairie : toutes bien documentées.

Pléthore d'icônes pensera-t-on, et que resterait-il de la normalisation de Microsoft ? Il resterait la logique rigoureuse de la programmation. Celle de savoir déterminer, sans se tromper, d'où vient l'erreur d'un programme qui ne produit pas ce que l'on attend exactement de lui. Alors que la méthode Microsoft, impose

d'être en permanence lié à l'éditeur pour demander des explications.

Et l'on arrive à Visual Basic Express 2005 : les formes **Drive, Dir, File** ont disparu, sauf si on les charge pour un projet ( ce que j'ai tenté de faire, en vain). Une autre méthode est disponible, qui doit faire la même chose, mais comment la mettre en oeuvre : point d'exemple en libre service qui dise exactement ce qu'il faut faire. Toujours la politique des petits morceaux qu'il faut chercher ici et là pour les réunir en bon ordre en un tout qui fonctionne.

Un constat s'impose : la technique des fenêtres (**Windows**), cache la liaison directe :

**Objet ==> Traitement ==> Objet résultat,**

Les utilisateurs ne voient que l'agrément visuels des fenêtres, ils en oublient ce qui est essentiel : comprendre comment se fabrique la transformation d'un objet, en un autre. Cette informatique du Visuel, telle qu'elle a été conçue et telle qu'elle évolue, détruit la compréhension des mécanismes de l'évolution.

Il ne reste qu'une informatique, qui paralyse l'intelligence et la créativité des humains. L'Ingénieur, par des méthodes recettes sans cesse répétées, produit du nouveau comme colonies d'insectes. Aucune vision de l'évolution n'est possible, l'A.D.N humain se fixe comme celui d'un insecte.

#### **Des deux dérives majeures.**

**Le concept de classe d'objet.** Les naturalistes, ont inventé une méthode de classement bien connue et basée sur l'usage du Latin. Un objet de la nature appartient à un groupe, s'il possède quelque chose que tous les membres du groupe ont en commun.

Cette propriété commune est codée, et le nom de l'objet, comprend obligatoirement ce code. Cela permet de faire des rapprochements.

Le tableau TF (B) obéit à cette règle.

Il eut été normal d'appliquer cette règle, en matière de langage objet. Plutôt que d'avoir des désignations qui ne parlent ni à l'esprit, ni à la lecture.

**Le concept de l'OS Windows.** L'exemple réel et parfait qui montre l'aberration du système. Prenons la cellule souche du tissu d'un organe, elle est à la base de la formation de l'organe. Mais cela fait, s'il fonctionnait comme Windows, chaque cellule de l'organe ne pourrait faire son travail qu'après quelle eut reçu son message de travail, et les données, par le canal obligatoire de la cellule souche. La vie n'aurait jamais pu évoluer par ce principe.

Ce ne sont que diverses preuves de la dérive de l'ordre informatique. Cette Informatique, malgré tous les avertissements donnés, produit in fine la destruction du cerveau d'un futur Diquipankù, pour conserver les privilèges acquis au bénéfice de l'élite dominante de la Terre, sous produit de ce cerveau. Et il devient celui d'un insecte, dont le critère d'évaluation se réduit à l'unité monétaire, qui est faite de deux seules dimensions L, T.