Méthode d'Analyse Structurée

NB : certaines notions présentée ici seront revues dans d'autres cours.

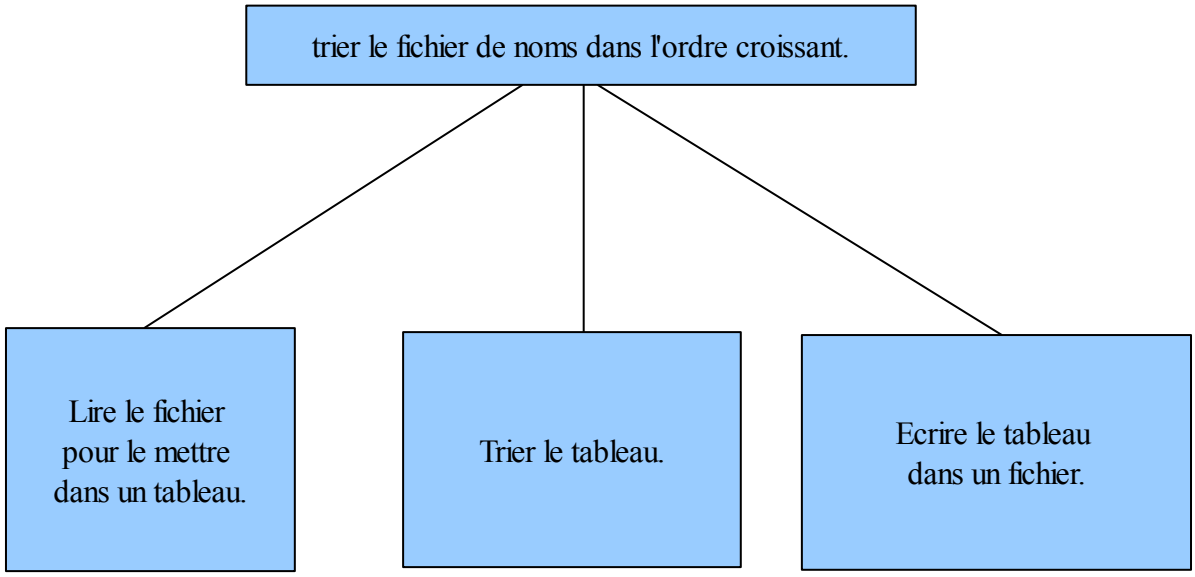
I. Introduction :

Nous avons un problème complexe à résoudre.

Par exemple : on veut trier un fichier contenant une liste de noms dans l'ordre alphabétique croissant.

La première idée est le principe : **« diviser pour régner ».**

Notre problème est trop compliqué essayons de le partager en plusieurs problème plus simples. L'algorithme résolvant un sous-problème s'appelle une **Procédure** (ou **fonction**).

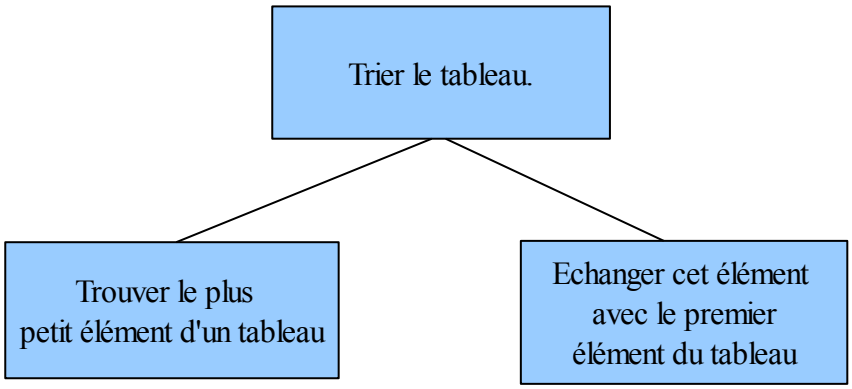


On peut ainsi écrire l'algorithme suivant :

Début
Lire le fichier pour le mettre dans un tableau.
Trier le tableau
Ecrire le tableau dans un fichier résultat.
Fin

Les 3 sous-problèmes peuvent eux mêmes être diviser en sous-problèmes.

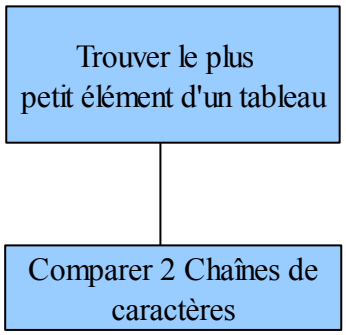
Pour le problème « trier un tableau », on peut par exemple le diviser en 2 sous-problème :



On peut donc écrire l'algorithme de la procédure (ou fonction) TrierTableau.

```
Var
   i : entier
Début
Pour i ← Premier Elément du tableau jusqu'au dernier élément du tableau faire
        Début
        entier iPPE;
        iPPE ← Trouver le plus petit élément du tableau à partir de i;
        Echanger les éléments i et iPPE du tableau;
        Fin
Fin
```

La fonction « trouver le plus petit élément du tableau » que l'on peut diviser en sous-problèmes.

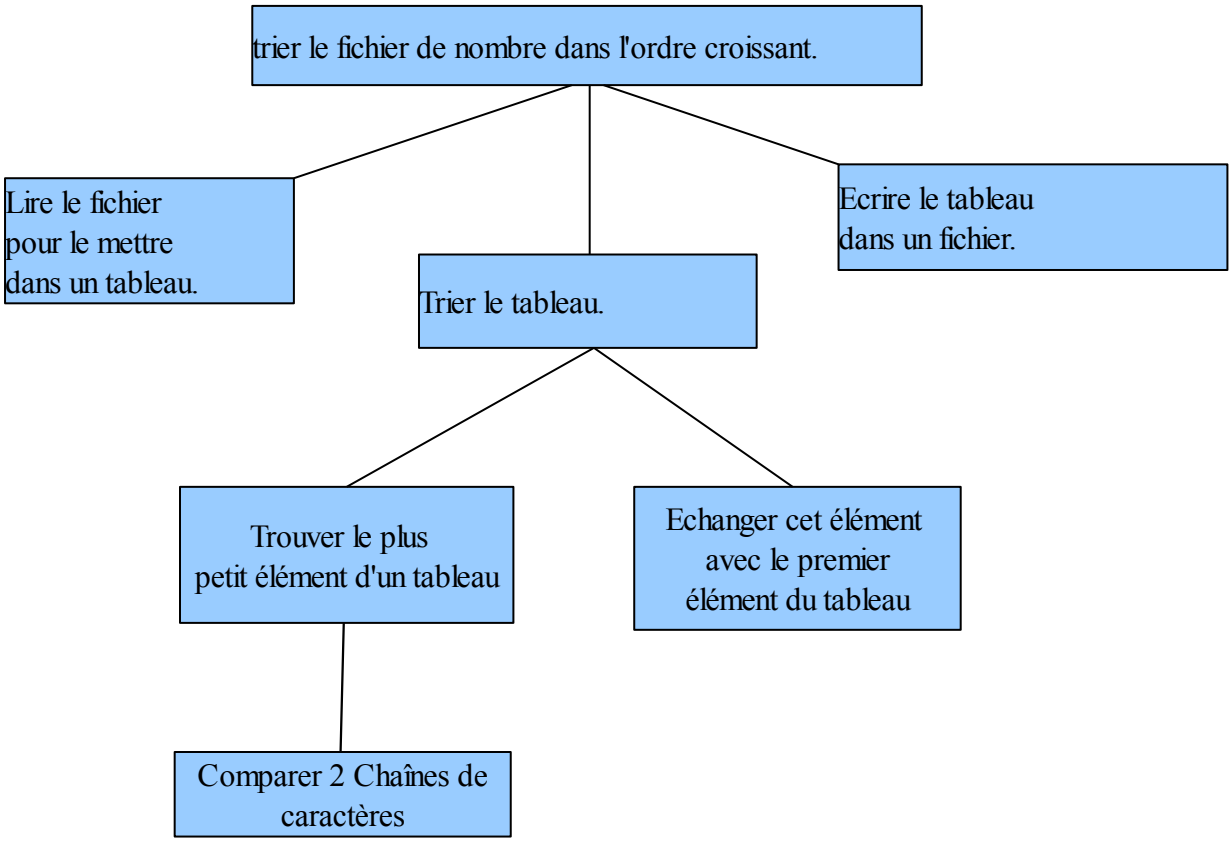



```
Début
iAux : entier
iC : entier
iAux ← i
Pour  iC ← i jusqu'à dernière élément du tableau
       début
       si  tableau[iChaîne] < tableau[iAux] alors iAux = iC
       Fin
Fin
```

II. Programmation Structurée Top/Down :

L'approche que nous venons d'utiliser s'appelle la méthode **fonctionnelle Top Down**. Cette méthode aboutis à une **structure** appelé arbre. Cette méthode conduit à une programmation structurée et est facilitée par l'utilisation de langage appelé Structuré (APL, Pascal, C, …) .

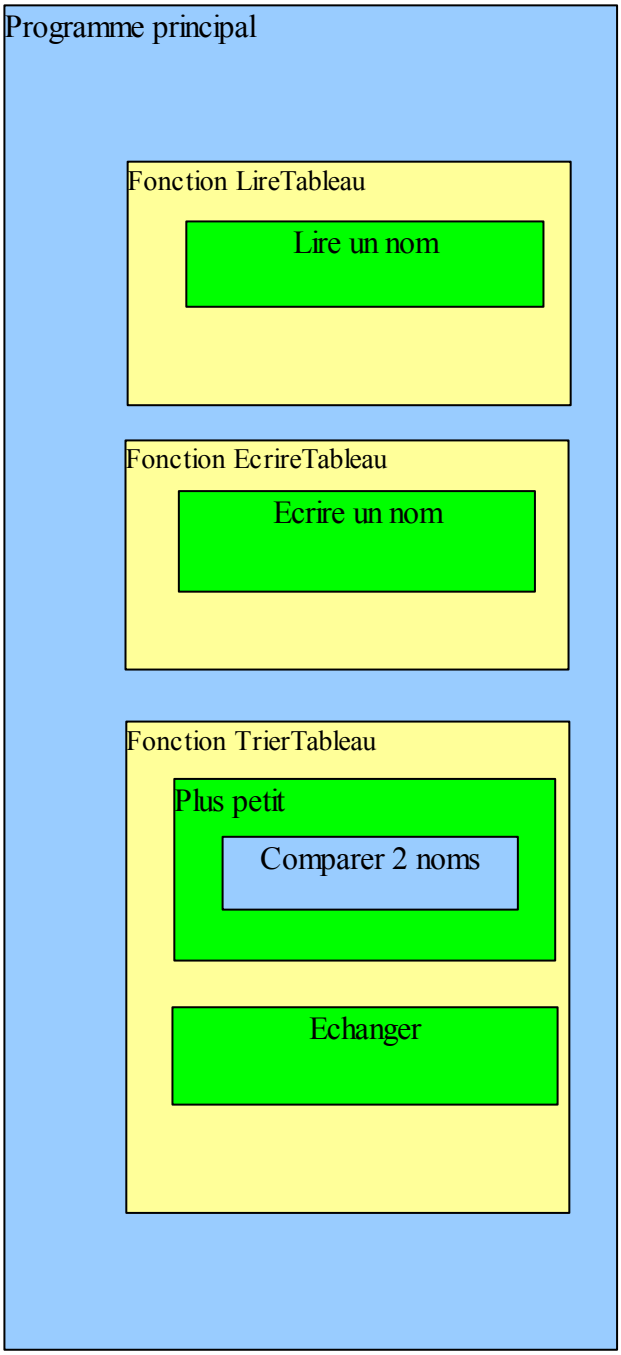
On peut résumé notre analyse par un arbre :



III. Imbrication des fonctions :

On peut aussi représenter l'analyse Structurée sous la forme représentant l'imbrication des fonctions. Ici il y a 4 niveaux de son imbrication.

NB : chaque fonction ne peut utiliser que les fonctions qu'elle contient et celle de même niveau d'imbrication. On définit ainsi une notion de **portée**. (pe « Trier Tableau » ne connait que « plus petit » et « Echanger ».



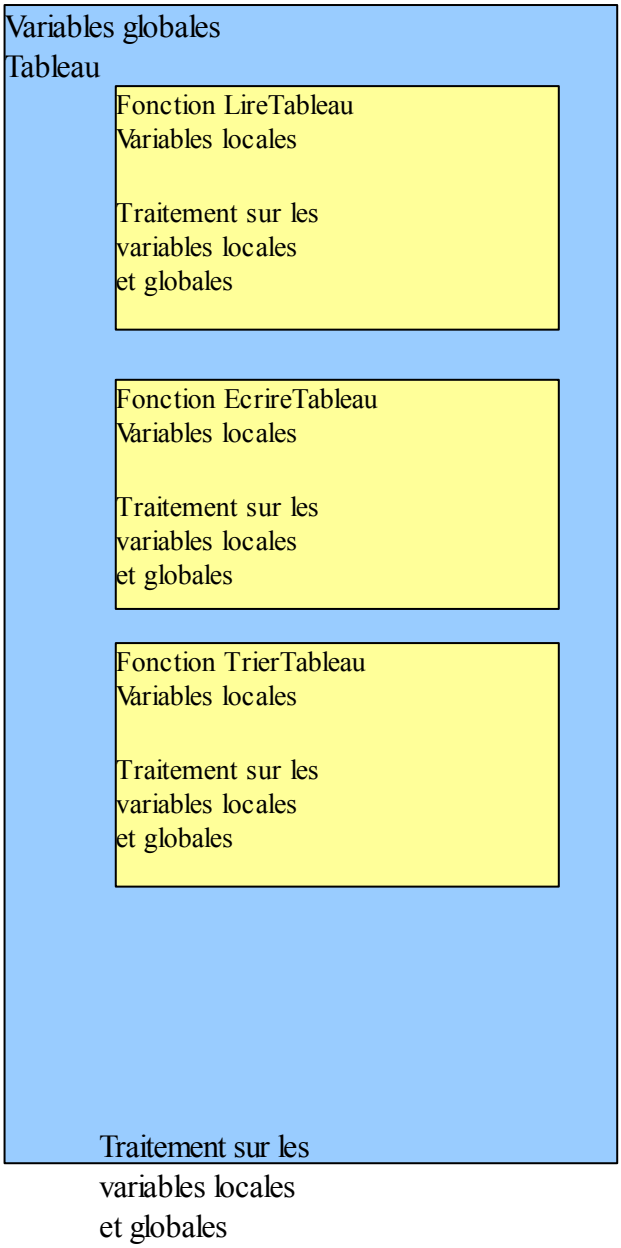
IV. Portée des variables :

La variable « tableau » peut-être utilisé par toutes les procédures du programme : c'est une variable globale à tout le programme.

La variable i de « TrierTableau » peut-être utilisé seulement par la fonction « TrierTableau » : c'est une variable locale à la fonction.

Définition : une variable globale est une variable qui est accessible dans tout le programme.
Définition : une variable locale est une variable qui n'est accessible que dans une partie du programme, à l'intérieur d'une fonction par exemple.

Structure d'un programme en langage C :



V. **Notion de module :** on regroupe les fonctions et les variables globales associés dans des fichiers sources appelés **modules**.

Quand un module veut utiliser une fonction ou une variable d'un autre module, il déclare un alias de la fonction ou du module en utilisant le mot « extern ».

Chaque module sera compilé (.obj) séparément et lié (link) ensemble pour former un exécutable (.exe).

Les modules peuvent être regrouper dans des librairies statiques (.lib) liées dans l'exécutable (.exe) ou dans des librairies dynamiques (.dll) liées à l'exécution du programme.