ALGO Ch 1 – Introduction

I. Définitions et exemples

A. A l'origine des algorithmes

La notion d'algorithme remonte à l'antiquité (*3 millénaires avant JC, chez les babyloniens, en Mésopotamie, actuellement l'Irak*) et décrivait des méthodes de résolution d'équations. Le mathématicien grec Euclide (*plus récemment, 3 siècles av. JC*) a décrit la détermination du plus grand diviseur commun de 2 nombres sous forme d'une suite de calculs.

Mais c'est bien plus tard qu'on a pu nommer ces démarches « algorithme ». C'est en effet du nom d'un mathématicien perse (800 ap. JC, actuel Ouzbékistan), Al-Khawarizmi, qu'on a introduit, 600 ans plus tard, vers 1550, le terme « algorithme ».

B. L'algorithme

Un ALGORITHME est un enchaînement ordonné d'actions élémentaires permettant l'obtention d'un résultat déterminé.

En d'autres termes :

C'est **l'expression de la résolution d'un problème posé**.

C'est **le moyen d'obtenir un résultat souhaité**.

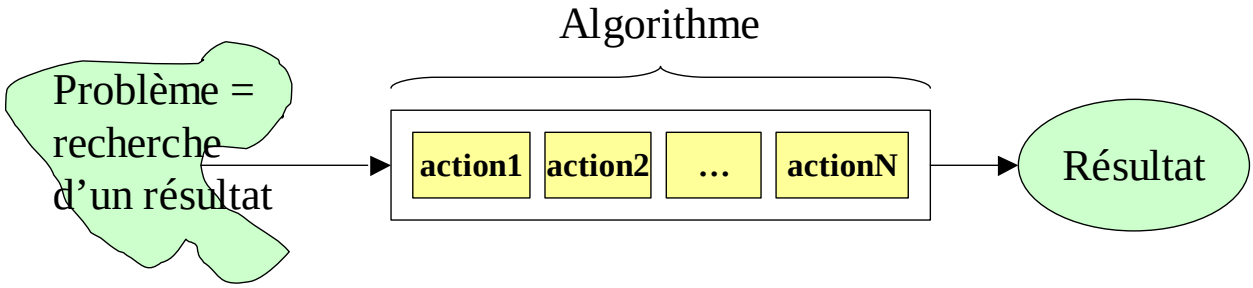
-La **succession des actions** forme les **étapes de la résolution du problème** posé initialement ←→ on ne peut concevoir un algorithme que si un problème est soluble !

Synonymes ou termes proches : procédure, mode opératoire, programme, recette, etc.

Chaque ACTION d'un algorithme modifie les éléments de détermination du résultat et représente une évolution nécessaire à l'élaboration de la solution à laquelle on doit aboutir.

Certaines actions sont élémentaires, non décomposables.

D'autres actions regroupent un certain nombre d'actions élémentaires et constituent des procédures ou fonctions.



C. L'algorithmique

L'ALGORITHMIQUE consiste en un ensemble de REGLES et de TECHNIQUES mis en œuvre dans la construction des algorithmes.

D. Le processeur

Les actions élémentaires d'un algorithme doivent être exécutées les unes à la suite des autres par un système de traitement (« un être humain » ou « une machine »).

Le **PROCESSEUR** est le système chargé d'exécuter l'algorithme : il devra savoir comment en exécuter chacune de ses actions élémentaires.

Le processeur des algorithmes est le plus souvent l'être humain.

Origine de processeur : anglais *to process* = traiter

E. Exemples d'algorithmes

Dans « la vraie vie », nous exécutons chaque jour des suites d'actions qui sont en fait des formes d'« algorithmes », le plus souvent inconsciemment.

Exemple : (résultat recherché : « la voiture est sortie du garage »)

Algorithme : « sortir la voiture du garage »

1. Aller jusqu'au garage
2. Ouvrir la porte du garage
3. Aller jusqu'à la voiture
4. Ouvrir la portière
5. Monter dans la voiture
6. Mettre la clef dans le contact
7. Démarrer le moteur
8. Passer la marche arrière
9. Reculer jusqu'à être à l'extérieur du garage
10. Arrêter le véhicule
11. Mettre au point mort
12. Arrêter le moteur
13. Retirer la clef du contact
14. Descendre de la voiture
15. Fermer la portière
16. Aller jusqu'au garage
17. Fermer la porte du garage
18. Aller jusqu'à la voiture

➔ Résultat = la voiture est sortie du garage (OK)

Exemple : (résultat recherché : un brownie prêt à déguster)

Algorithme : « préparer un brownie »

1. Mélanger le sucre semoule, le sucre vanillé, les œufs et la farine tamisée
2. Faire fondre le beurre
3. Mettre le four à préchauffer

et pendant ce temps :

4. Mélanger le beurre à la pâte jusqu'à avoir un mélange homogène
5. Faire fondre le chocolat
6. Mélanger le chocolat à la pâte tant que le mélange n'est pas homogène
7. Mélanger les noix de Pécan et la poudre d'amande à la pâte
8. Verser la pâte dans un moule à gâteau beurré
9. Mettre à cuire pendant 35 minutes dans le four préchauffé à 170°C
10. Arrêter le four
11. Sortir le gâteau

➔ Résultat = le brownie est prêt (OK)

L'exécution des actions permettant la réalisation de la recette est **séquentielle** : les actions sont réalisées les unes à la suite des autres.

Certaines actions sont également répétées jusqu'à ce qu'elles soient complètement réalisées.

D'autres encore sont amenées à se dérouler en parallèle.

F. Environnement d'un algorithme

La suite d'actions constituant un algorithme nécessite très souvent l'utilisation d'éléments extérieurs, indispensables à l'exécution des actions élémentaires :

- Les ingrédients dans le cas d'une recette,
- Le détail des salaires et dépenses pour la vérification d'un relevé de compte,

Un algorithme nécessite très souvent la fourniture d'éléments (« en entrée ») pour produire un résultat (« en sortie ».)

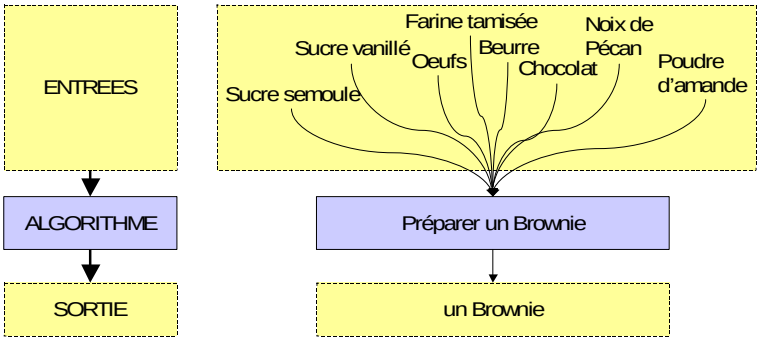


Figure 1 : l'algorithme "Préparer un Brownie" : transforme les ingrédients qui lui sont fournis (=en entrée) pour produire un résultat, le Brownie (=en sortie)

G. Caractéristique essentielle d'un algorithme

Un ALGORITHME devra fournir un RESULTAT IDENTIQUE POUR LES MEMES ELEMENTS FOURNIS EN ENTREE, dans des conditions de fonctionnement identiques, ceci EN UN TEMPS DETERMINE.

I. Algorithmique et mathématique <new>

A. Fonctions numériques

Les fonctions numériques mathématiques représentent une première forme d'algorithmes calculatoires.

A travers la définition formelle d'un ensemble de départ et d'un ensemble d'arrivée, et d'une règle de transformation, la fonction.

II. Algorithmique et informatique

A. Algorithmes informatiques

Un ALGORITHME INFORMATIQUE est un enchaînement d'instructions élémentaires nécessaire à la résolution de problèmes généralement calculatoires (= mettant en œuvre des calculs)

Les éléments qui lui seront fournis en entrée et le résultat produit seront maintenant des <u>données</u> (représentations codées du monde réel) plutôt que des <u>objets</u> du monde réel.

On peut bien sûr étendre la « notion d'algorithme calculatoire » à toutes les formes de pilotage de machines et robots qui nécessitent des calculs complexes et appliqués à des données fournies par des capteurs sous forme de signaux numériques.

B. Exemples

Exemples d'algorithmes calculatoires « de la vraie vie » :

<u>Algorithme : « calculer le coût d'un crédit »</u>

<u>Données variables :</u> montant emprunté, remboursement mensuel, nombre de mois

1. Calculer le montant total remboursé = remboursement mensuel X nombre de mois
2. Calculer le coût du crédit = montant total remboursé – montant emprunté

➔ Résultat = coût du crédit

Algorithme : « calculer la surface de papier peint à poser dans une pièce rectangulaire »

Données en entrée : hauteur de la pièce, largeur et longueur de la pièce, nombre de portes, nombre de fenêtres

Données intermédiaires :

Données constantes : surface d'une porte : 2 m2, surface d'une fenêtre : 2 m2

1. Calculer la surface totale des murs = ((largeur + longueur) X 2) X hauteur
2. Calculer la surface des portes = surface d'une porte X nombre de portes
3. Calculer la surface des fenêtres = surface d'une fenêtre X nombre de fenêtres
4. Calculer le surface de papier peint = surface des murs – surface des portes – surfaces des fenêtres

➔ Résultat = surface de papier peint

Et pour reprendre d'algorithme élaboré par Euclide :

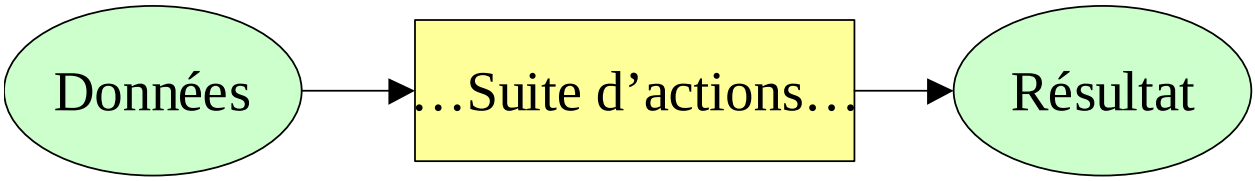
Algorithme : « calculer le Plus Grand Commun Diviseur de 2 nombres »

Données en entrée : nombre1 et nombre2, non nuls et nombre1 > nombre2

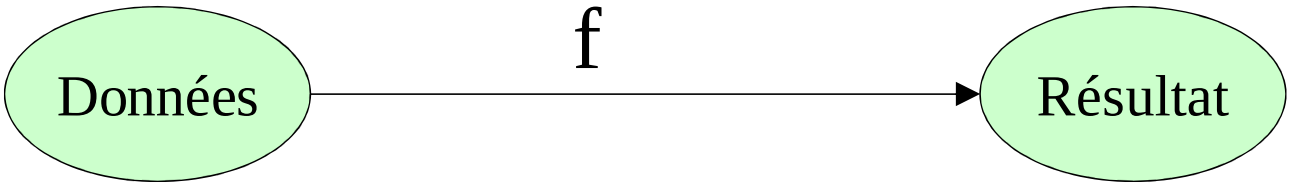
Données intermédiaires : reste

1. Calculer le reste de la division de nombre1 par nombre2
2. Tant que le reste est différent de 0
3. nombre1 prend la valeur de nombre2
4. nombre2 prend la valeur de reste
5. Calculer le reste de la division de nombre1 par nombre2

➔ Résultat = valeur de reste



Dans le traitement des informations, on peut représenter l'algorithme comme une fonction f, de D (les données) dans R (les résultats possibles) :



C. De l'algorithme informatique au programme informatique

Pour qu'un algorithme (processeur = être humain) puisse être exécuté par un ordinateur, il est nécessaire d'en effectuer la traduction dans un langage compréhensible par celui-ci (*langage machine*).

Un PROGRAMME INFORMATIQUE est une succession d'instructions écrites en langage machine et exécutables par un ordinateur, afin de traiter les données d'un problème et de renvoyer un résultat à l'utilisateur du programme.

Le terme « Logiciel » est souvent associé à un ensemble de programmes permettant de résoudre une catégorie de tâches.

Le langage machine (*dit « langage de bas niveau »*) étant complexe et délicat à écrire, les chercheurs ont inventé des langages plus simples à manipuler pour l'être humain, à mi-chemin entre la forme algorithmique (proche de la langue naturelle) et le langage machine : ce sont les langages de programmation (*dits « langages de haut niveau »*).

L'activité de CODAGE ou PROGRAMMATION INFORMATIQUE consiste à traduire un algorithme en « programme source » exprimé dans un langage « cible » (celui vers lequel on traduit), généralement langage de haut niveau comme C, C++, PHP, Java, Logo, Visual Basic, etc. (plus d'une centaine de langages informatiques).

La construction de l'algorithme est donc une des premières tâches en programmation et sera indispensable pour produire des programmes de qualité.

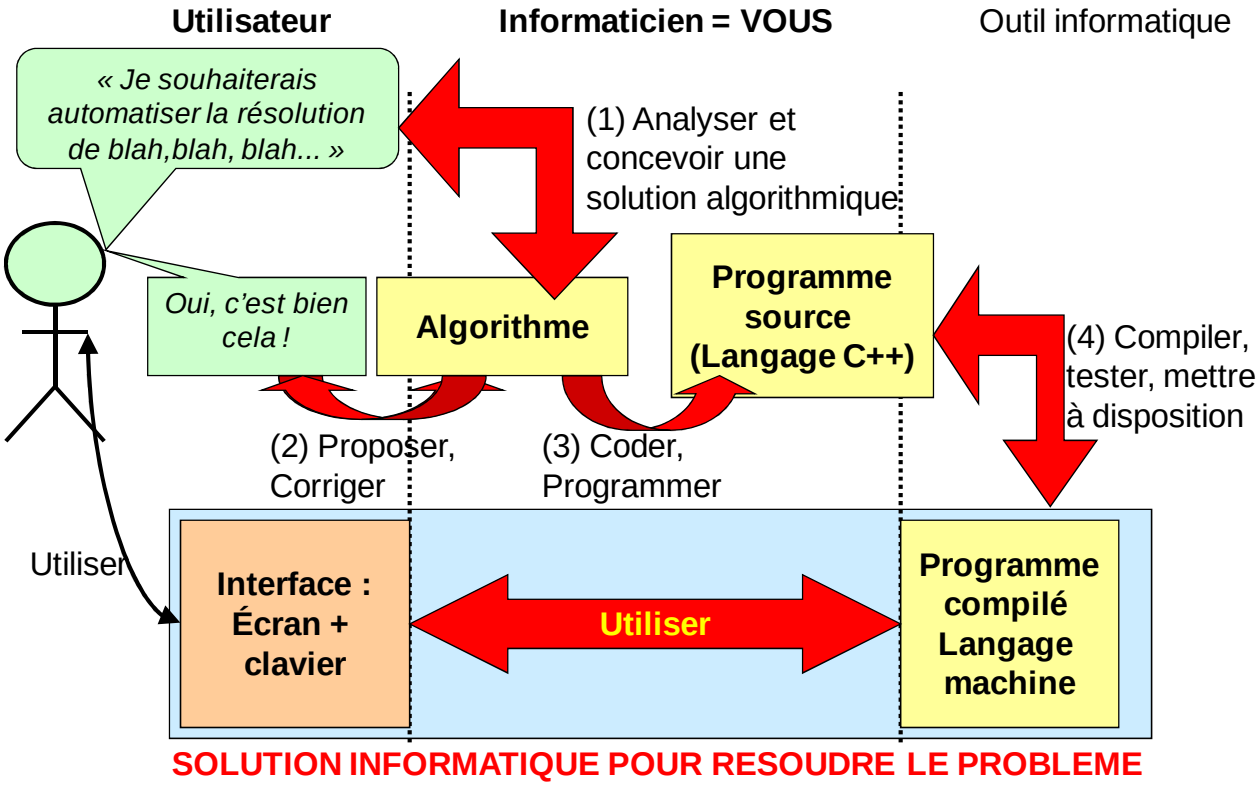


Figure 2 : les étapes de construction d'un programme informatique – vision simplifiée

D. Des besoins des utilisateurs à l'algorithme

La construction d'un algorithme n'est pas la première étape du processus de développement d'applications.

1. En effet, il est d'abord indispensable de bien comprendre les besoins réels de l'utilisateur afin d'établir un **cahier des charges** (expression exhaustive des besoins de l'utilisateur, définis dans un langage naturel)
2. Ce cahier des charges va ensuite être transformé en une **spécification des besoins**, document précisant dans un langage d'informaticien « ce qu'il faudra développer », pour répondre aux besoins exprimés dans le cahier des charges.
3. La phase suivante sera la conception de **l'algorithme** permettant de définir le « comment développer ce qui est demandé », en utilisant 2 axes d'analyse :
    a. Analyse descendante : décomposition d'un problème de complexité inférieure ; si les sous-problèmes paraissent encore trop complexes, ils sont décomposés à leur tour, etc.
    b. Analyse ascendante : composition d'une solution à partir de sous-algorithmes déjà définis (et fiables)

La vision simplifiée, exprimée par la figure 2, a pour objectif de montrer la nécessité de « définir, valider, décrire », avec pour aboutissement la production d'un algorithme, **avant** de « coder », de se lancer dans l'écriture du programme source dans un langage de programmation.

E. Démarche pour l'écriture d'un algorithme

L'écriture d'un algorithme est l'aboutissement d'une phase d'analyse dont la méthode dépend souvent du problème posé. On peut donner 2 grandes étapes :

1. Analyser les données

Recenser toutes les données au sens concept ; chaque concept doit être représenté par un objet qu'il faut judicieusement nommer pour l'identifier ; il faut ensuite distinguer les objets constants et variables ; puis il faut classer les objets pour distinguer ceux qui vont stocker les valeurs de base, intermédiaire et résultat.

Pour recenser de manière exhaustive les objets, une méthode simple consiste à partir du résultat et à remonter aux données de base par l'intermédiaire des formules de calcul.

2. Structurer l'algorithme

Chercher la structure générale de l'algorithme : en général, on y trouve (parfois regroupées)

- une phase d'**initialisation** de variables, par affectation de valeurs, ou par lecture de l'environnement (par exemple, une saisie d'un utilisateur),
- une phase de **traitement**, dans laquelle les données sont transformées pour aboutir au résultat final,
- une phase de fourniture du **résultat** attendu, par retour d'une valeur ou par écriture vers l'environnement (par exemple, un affichage sur un écran) ; cette phase est parfois intégrée à la phase précédente, des résultats intermédiaires apparaissant au fur et à mesure du traitement

3. Module de contrôle et modules fonctionnels

Un algorithme complexe comportera très souvent :

- un ensemble de modules fonctionnels (des fonctions) définissant les fonctions élémentaires assurés
- un module de contrôle de la séquence d'exécution des modules fonctionnels (la fonction principale)

III. Notations algorithmiques

A. Notations algorithmiques

Une **NOTATION ALGORITHMIQUE** est une représentation simplifiée (= un langage) permettant de définir sans ambiguïté un algorithme informatique de manière à ce qu'il soit lisible par tous.

C'est une convention qu'on établit afin de pouvoir dialoguer avec le demandeur du programme. Il existe plusieurs notations algorithmiques.

« Sans ambiguïté » = sans qu'il y ait de doute, une seule interprétation possible

Notation : manière d'écrire, de représenter

Convention = règles sur lesquelles on se met d'accord (c'est une forme de contrat)

Les **PRIMITIVES** d'un langage algorithmique sont les instructions de base à partir desquelles on pourra construire toutes sortes d'algorithmes.

B. Le pseudo-code

Le **LANGAGE ALGORITHMIQUE**, ou **PSEUDO-CODE**, est la notation algorithmique la plus utilisée. Il définit un langage à mi-chemin entre le langage naturel et les langages de programmation, sans cependant tomber dans les particularités d'un langage de programmation spécifique.

Exemple d'algorithme exprimé en pseudo-code :

```
% à partir de la saisie de votre âge, cet algorithme détermine si vous êtes majeur %

ALGO TesterMajorite
ROLE Déterminer majeur ou pas en fonction de l'age
DECLARATION
CONSTANTES
     AGE_MAJORITE = 18
VARIABLES
     Vage : ENTIER

% actions à réaliser sur les données : CORPS  %
DEBUT
    AFFICHER ("donnez votre âge")
    SAISIR(Vage)
    SI (Vage >= AGE_MAJORITE)
       |  ALORS
       |      AFFICHER ("majeur")
       |  SINON
            AFFICHER ("mineur")
    finSI
FIN
```

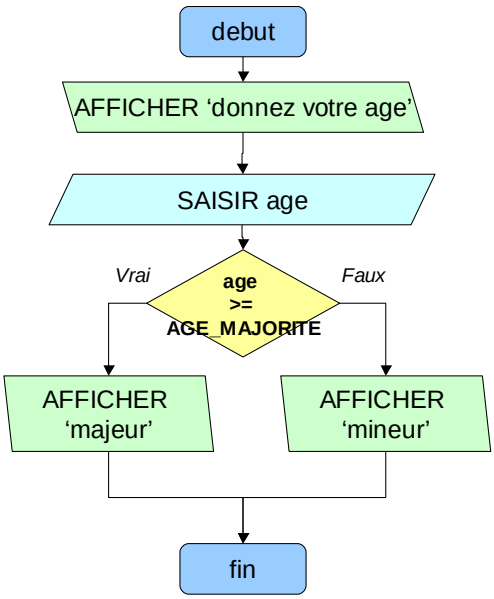
Attention :

Les langages algorithmiques ne sont pas standardisés : vous trouverez donc des variantes dans l'écriture des algorithmes, mais celles-ci ne devraient pas être un obstacle à la compréhension.

C. L'organigramme (ou ordinogramme)

L'organigramme est une représentation graphique utilisant des symboles visuels pour représenter la suite d'actions élémentaires d'un algorithme.

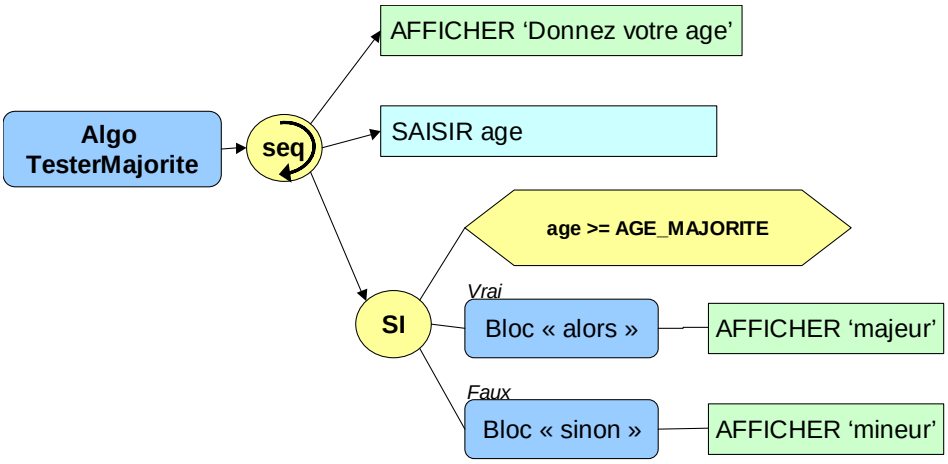
Exemple d'algorithme sous forme d'organigramme :



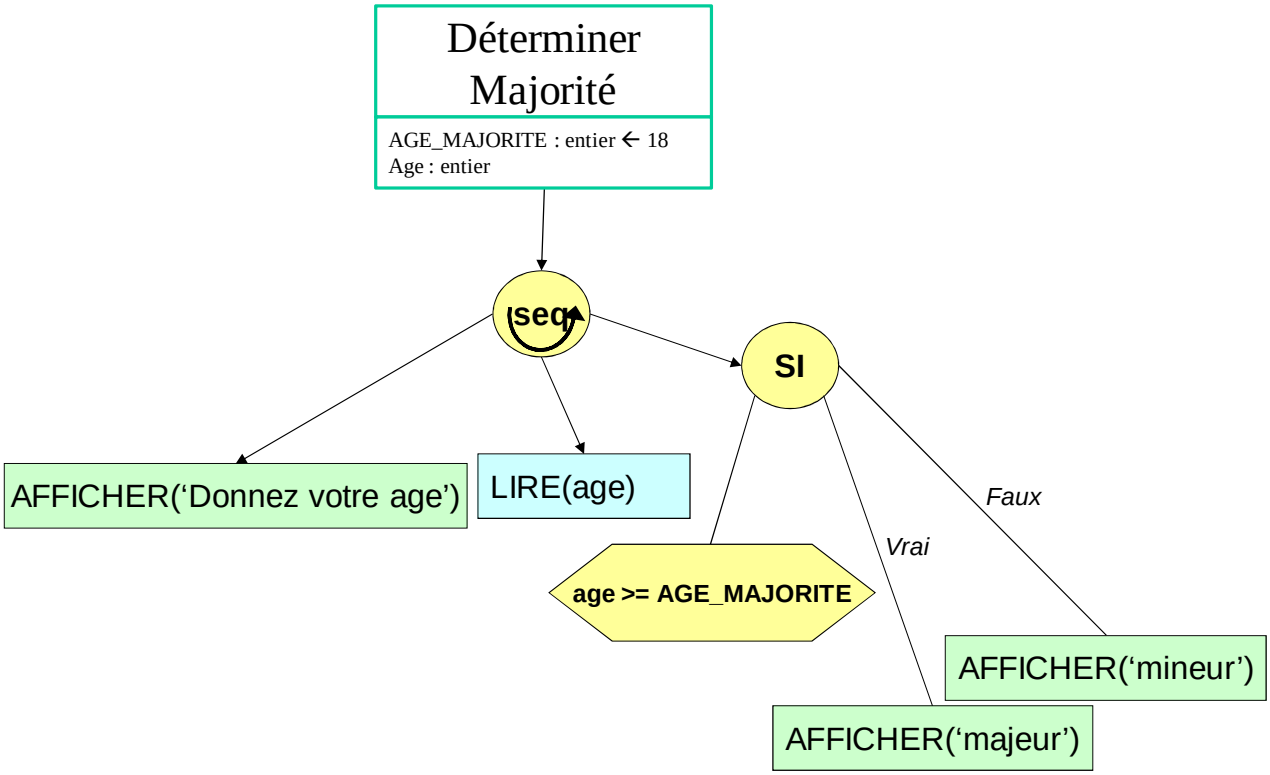
D. L'arbre programmatique

L'arbre programmatique est une évolution de l'organigramme et se rapproche du pseudo-code.

Exemple d'algorithme sous forme d'arbre programmatique (1) :



Exemple d'algorithme sous forme d'arbre programmatique (2)



D'autres représentations, comme le Grafcet, sont utilisés pour représenter les étapes de traitement des automates.

IV. Conclusion

Pourquoi donc utiliser encore un nouveau langage ???

L'utilisation d'un langage algorithmique (ou autre notation graphique) pour exprimer les algorithmes part de plusieurs nécessités :

- **Réfléchir** à la conception d'une solution dans une langue proche de notre langue naturelle ou sous une forme visuelle accessible à tous
- Pouvoir ainsi **Dialoguer** avec l'utilisateur demandeur
- **Préparer** l'activité de codage informatique (programmation dans un langage informatique),
- **Libérer** des particularités des langages informatiques (on peut très bien à avoir à demander le codage d'un algorithme à plusieurs programmeurs dans des langages informatiques différents)

L'expression graphique d'un algorithme est intéressante si sa taille est relativement petite.

Dès lors que ta taille de l'algorithme augmente, l'espace nécessaire à la représentation des symboles graphiques devient trop important et la lisibilité s'en trouve diminuée.

C'est pourquoi la notation en **PSEUDO CODE** est généralement privilégiée aujourd'hui pour représenter les algorithmes informatiques.