Algorithme

Chapitre 1

I. Phases d'élaboration d'un programme

Un programme avant son introduction en production (devient opérationnel) passe par 4 phases :

- Phase d'analyse
- Phase de spécification
- Phase de traduction
- Phase d'exécution

Phases pour aboutir à un programme exécutable présentable à un client.

Enoncé du problème -> *Analyse* -> **Modèle Conceptuel** -> *Spécification* -> **Algorithme** -> *Traduction* -> **Programme** -> *Exécution* -> **Réalisation**

Phase conceptuelle – Phase de réalisation
Données - *Phases*

1. Phase d'analyse

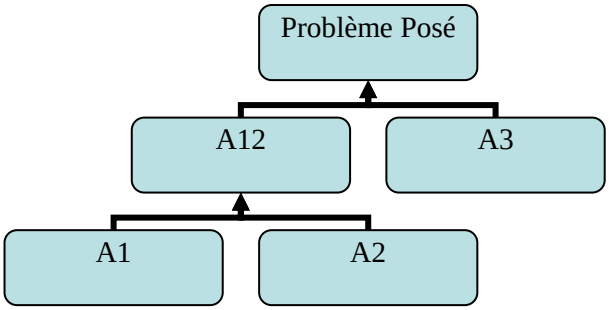
A partir de l'énoncé du problème (cahier de charges), la phase d'analyse consiste à élaborer le modèle conceptuel *MCA* qui devient la solution du problème dans un niveau sémantique trop élaboré (non précis) sans forme d'une suite d'étapes et d'actions.

Le *MCA* est une approche de la solution ne respectant aucun formalise (pas d'étapes d'écriture à respecter). **C'est un brouillon de solutions qui nous aide à trouver la solution du programme.**

Deux méthodes sont utilisées dans l'analyse de l'énoncé des problèmes.

a. La méthode ascendante.

A partir d'algorithmes élémentaires constituant une boite à outils, le concepteur construit la solution du problème. Le résultat est un *MCA* ascendant sous forme d'un arbre où les feuilles sont les algorithmes et les nœuds de l'arbre sont les étapes.

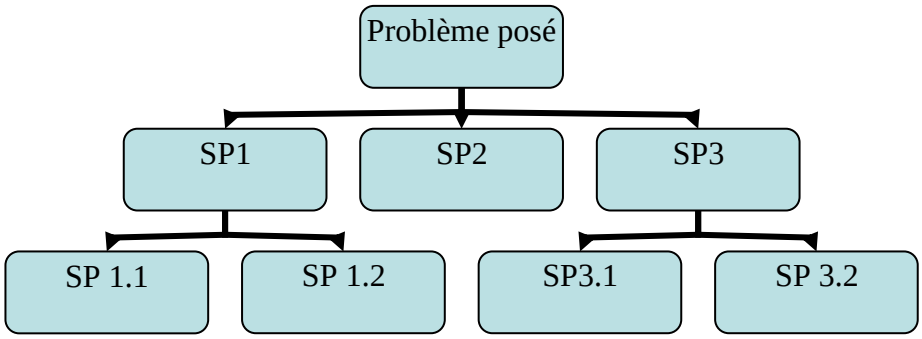


Tout cela est ASCENDANT !
A1, A2, A3 : Algorithmes élémentaires.
A12 : Nœud de l'arbre, c'est une étape.

Cette méthode exige du concepteur une expérience confirmée et une maitrise des outils pour qu'il puisse arriver à la solution du problème.

b. La méthode descendante.

Le concepteur subdivise le problème posé en plusieurs sous problèmes. Si ces derniers n'ont pas de solution facile ou connue, on les subdivise de nouveau. On obtient alors un *MCA* sous forme d'arbre avec des niveaux de difficultés.
Les feuilles de l'arbre sont les actions à réaliser et les nœuds sont les étapes.

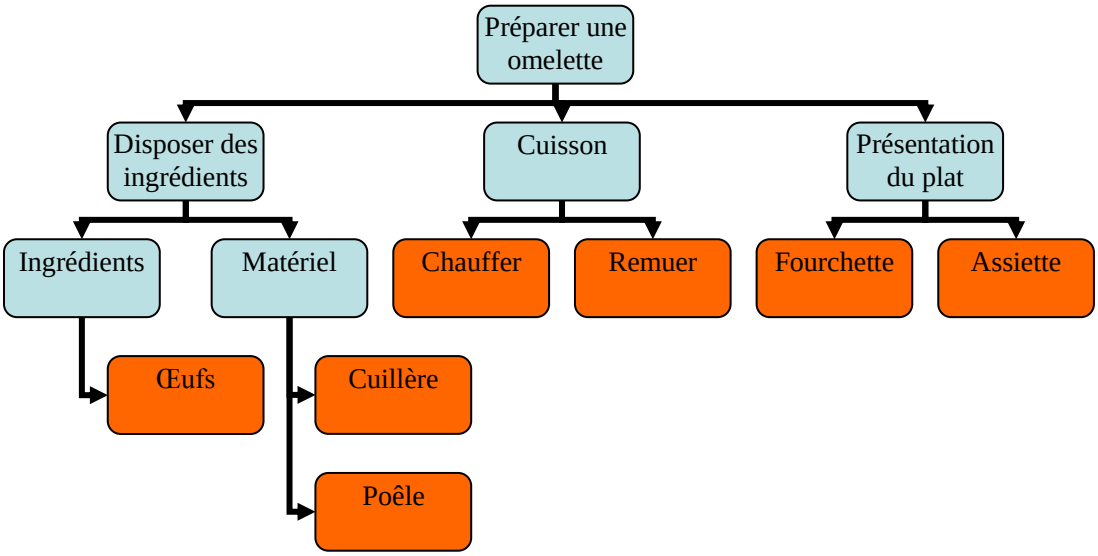


SP1 et SP3 sont les nœuds de l'arbre.

SP1.1, SP1.2, SP2, SP3.1 et SP3.2 sont les feuilles de l'arbre qui donnent la solution au problème.

Exemple 1 :

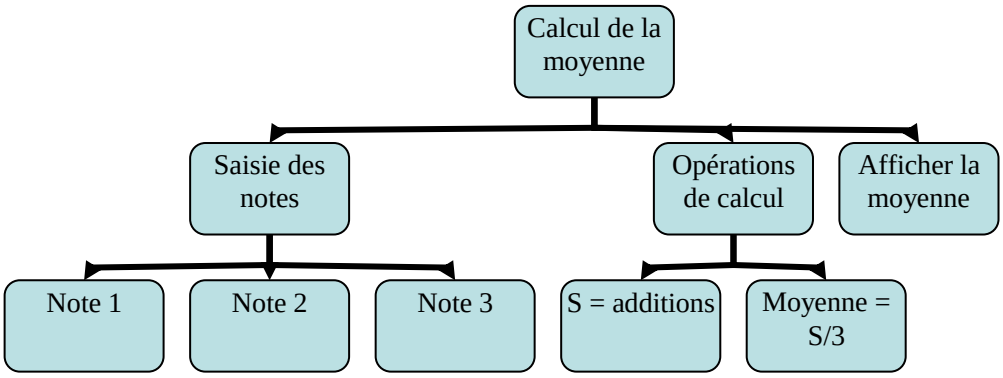
Elaborer un *MCA* qui permet de modéliser la préparation d'une omelette.



La solution du problème est donnée par la liste des feuilles de l'arbre.

Exemple 2 :

Elaborer un *MCA* qui permet de calculer la moyenne de trois notes.



La solution du problème est donnée par la suite des feuilles de l'arbre, à savoir :

- Saisie note 1
- Saisie note 2
- Saisie note 3
- S = note1 + note2 + note3
- Moyenne = S/3
- Afficher Moyenne

2. La phase de spécification

La phase de spécification consiste à produire un algorithme à partir du modèle conceptuel d'analyse en spécifiant (rendre précis) les actions et les étapes de ce modèle. La production d'un algorithme nécessite le respect du formalisme algorithmique (ensemble de règles syntaxiques et sémantiques). Un algorithme est une suite finie d'actions et d'étapes donnant solution à un problème posé.

Exemple d'écriture d'actions.

MCA	***Algorithme***
Saisir note 1	Saisir (note1)
Afficher la moyenne	Afficher (« La moyenne est de, moyenne »)

3. La phase de traduction

Pour exécuter un algorithme sur une machine on doit le traduire dans un langage évolué compréhensible par la machine (Java, C, C++,…).
Pour se faire la maitrise du formalisme du langage est obligatoire.

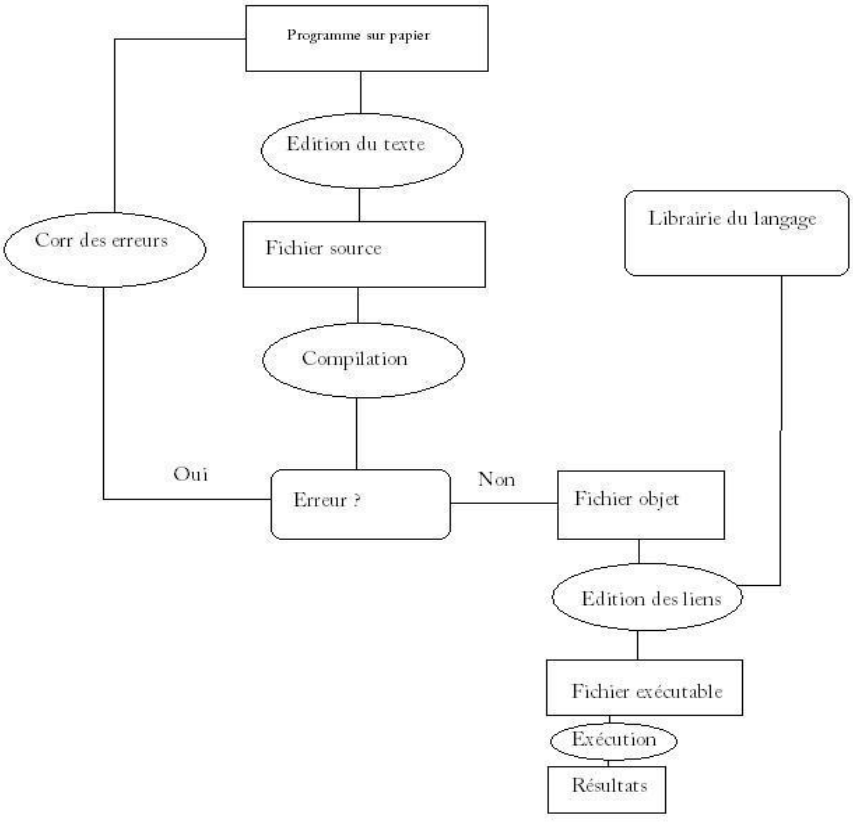
Exemple de traduction.

Algorithme	C	C++
Saisir (note1) Afficher (« La moyenne est de,moyenne »)	scanf (« %f »,&note1) ; Printf(« La moyenne est de : %f »,moyenne) ;	Cin>>note1 ; Cont<< « La moyenne est de : »<<moyenne ;

4. La phase d'exécution.

La dernière étape qui s'effectue devant une machine, elle est constituée des 4 étapes suivantes :

- **Edition de texte** : la saisie du programme.
- **Compilation** : la recherche d'éventuelles erreurs syntaxiques et sémantiques.
- **Edition des liens** : inclusion des codes de librairie du langage.
- **Exécution du programme** : obtention des résultats.



II. Syntaxe générale.

Un algorithme est défini par son nom et est composé de deux parties :

- Une partie déclaration : où sont définis les objets de données de l'algorithme.
- Une partie corps : où sont regroupées les actions qui agissent sur les objets de données.

```
Algo : nom_algo
Déclaration
|
|Objet de données
|
Début
|
| Liste des actions
|
Fin nom_algo
```

Les commentaires :

Pour documenter un algorithme, on insère des commentaires.
Un commentaire est inséré entre (* et *) ou il débute par // s'il ne tient que sur une seule ligne.

III. Objet de données et types.

Un objet de données et défini par un nom, un type et une nature.

Le nom : est un identificateur unique composé de suite de lettres, chiffres et caractères.

A utiliser	A ne pas utiliser
Annee	Année
Age	L'âge
Prix_HT	Prix HT
Tour33	33 Tour

On doit utiliser des noms d'objets de données explicites pour faciliter la lecture du code.

Le type : défini l'ensemble des valeurs que peut prendre un objet de donnée.

Type	Description
Entier	Valeur + ou - € Z
Réel	Valeur + ou - € R
Caractère	Lettre, Chiffre, Symbole

Chaine (taille)	Suite de caractères avec une taille
Boléen	Deux valeurs -> Vrai ou Faux

La nature : un objet de donnée peut être variable (change de contenu) ou constante (garde le même contenu).

Déclaration d'une variable :

nom_variable : type

Ex : Prix : réel
Heure : entier

Déclaration d'une constante :

nom_constante = valeur

Ex : Constante tva=19.6

IV. Les actions.

a. La saisie

Pour affecter une valeur saisie pour l'utilisation à une variable, on utilise l'une des deux fonctions : Lire ou Saisir.

Syntaxe :
- Saisir (nom_variable)
- Lire (nom_variable)
- Saisir (nom_variable1,nom_variable2,…)
- Lire (nom_variable1,nom_variable2,…)

Ex :

Prix : Réel	h,m,s : entier
Saisir(Prix)	Saisir(h,m,s)

b. L'affichage

Pour afficher un message ou le contenu d'une variable à l'écran, on utilise l'une des deux fonctions : Afficher ou Ecrire.

Syntaxe :
- Afficher (« message »)
- Afficher (nom_variable)
- Afficher (« message : » ,nom_variable,…)

Ex :

Moyenne : Réel Afficher (« La moyenne est de : », moyenne)

V. Les opérations

a. Opérations arithmétiques

Opération	Description	Exemple
+	addition	x + y
-	soustraction	x – y
*	multiplication	x*y
/	division réelle sans reste	x/y
Div :	division entière avec reste	x Div y x : y
Mod %	Modulo, reste de la division	x Mod y x%y

b. Opérations de test

Opération	Description	Exemple
>	Supérieur à	x>y
>=	Supérieur ou égal à	
<	Inférieur à	x<y
<=	Inférieur ou égal à	
=	Egal à	x=y
!=	Différent de	x !=y

c. Opérations logiques

Opération	Description	Exemple
Non	Négation	Non(x)
ET	ET logique injonction	x E y
OU	OU logique inclusif disjonction	x OU y

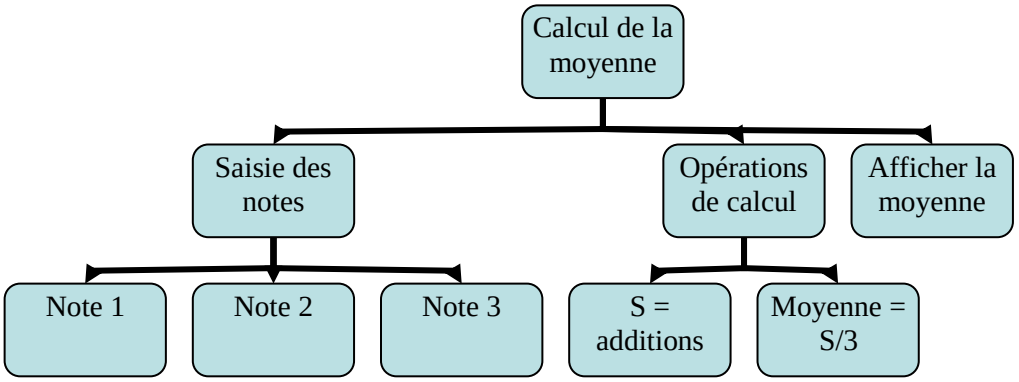
Table de vérités :
0 → Faux
1 → Vrai

x	Non(x)
0	1
1	0

x	y	x ET y	x OU y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

VI. Exemple d'écriture d'algorithme

Reprends l'exemple 2 du cours.



Algorithme : calcul_moyenne

```
Déclaration
|
| note1, note2, note3, S, Moyenne : réel
|
Début
|Afficher (« donner la note1 : »)
|Saisir (note1)
|Afficher (« donner la note2 : »)
|Saisir (note2)
|Afficher (« donner la note3 : »)
|Saisir (note3)
|S ← note1+note2+note3
|Moyenne ← S/3
|Afficher (« La moyenne est de : », Moyenne)
Fin calcul_moyenne
```