Chapitre 16

INFORMATIQUE ET AUTOMATISME

Algorithmie

1

1 - PREAMBULE

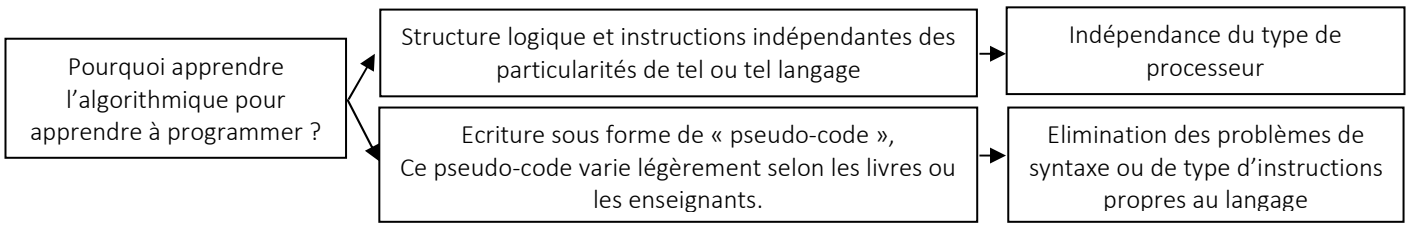
Un processeur d'une machine informatique n'est pas capable de faire autre chose que de suivre des instructions les unes derrière les autres (un peu à l'image d'une recette de cuisine ou d'un mode d'emploi d'un appareil).

Le meilleur moyen pour décrire son comportement est de travailler sous forme d'algorithmes.

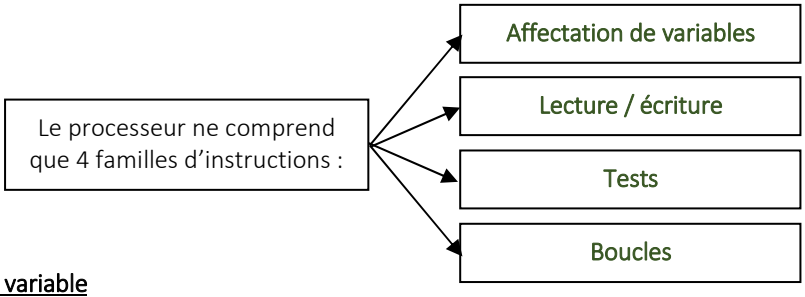
La maîtrise de l'algorithmique requiert deux qualités, très complémentaires :

- Il faut avoir une certaine intuition, car aucune recette ne permet de savoir a priori quelles instructions permettront d'obtenir le résultat voulu. Ce qu'on appelle l'intuition n'est finalement que de l'expérience tellement répétée que le raisonnement, au départ laborieux, finit par devenir « spontané » ;
- Il faut être méthodique et rigoureux. En effet, chaque fois qu'on écrit une série d'instructions qu'on croit justes, il faut systématiquement se mettre mentalement à la place de la machine qui va les exécuter, armé d'un papier et d'un crayon, afin de vérifier si le résultat obtenu est bien celui que l'on voulait. Cette opération ne requiert pas la moindre once d'intelligence. Mais elle reste néanmoins indispensable, si l'on ne veut pas écrire à l'aveuglette.

2 - ALGORITHME OU PROGRAMME ?



3 - QUATRE FAMILLES D'INSTRUCTIONS, ET C'EST TOUT !



Les différents types de variable

Une variable est une zone mémoire réservée pour stocker le résultat d'une opération

Type		Plage	Espace utilisé en mémoire
Booléen	Boolean (TOR)	VRAI (TRUE – « 1 ») ou FAUX (FALSE – « 0 »)	1 bit
Numérique	Byte (octet)	0 à 255 ou -128 à +127	1 octet
	Entier simple	-32 768 à 32 767	2 octets
	Entier long	-2 147 483 648 à 2 147 483 647	4 octets
	Réel simple	$-3{,}40\times10^{38}$ à $-1{,}40\times10^{-45}$ pour les valeurs négatives $1{,}40\times10^{-45}$ à $3{,}40\times10^{38}$ pour les valeurs positives	4 octets
	Réel double	$-1{,}79\times10^{308}$ à $-4{,}94\times10^{-324}$ pour les valeurs négatives $4{,}94\times10^{-324}$ à $1{,}79\times10^{308}$ pour les valeurs positives	8 octets
Alphanumérique	Caractère	Tout type de caractère (lettres, signes de ponctuation, espaces, chiffres), code ASCII	1 octet
	Chaine	Ensemble de caractères les uns derrière les autres En pseudo-code, une chaîne de caractères est toujours notée entre guillemets	fonction du nombre de caractères contenus dans la chaine
« Assemblage »	On peut assembler des types de variable de base pour se créer son propre type (date, …)		

L'affectation

La seule chose qu'on puisse faire avec une variable, c'est **l'affecter,** c'est-à-dire **lui attribuer une valeur**.

En pseudo-code, l'instruction d'affectation se note avec le signe ←

Ainsi :	Variable Toto	Variable Tutu
Toto ← 24	24	xx
Tutu ← Toto + 4	24	28

Une instruction d'affectation ne modifie que ce qui est situé à gauche de la flèche.

Opérateurs possibles dans les expressions :

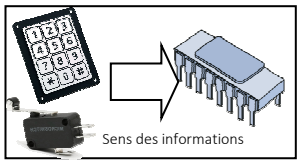
Variable	Opérateurs
Booléen	ET, OU, NON, XOR
Numériques	+, -, /, *, (,),^
Chaine	& (*concaténer*)

La lecture / l'écriture.

Le processeur peut avoir besoin de communiquer avec l'extérieur

LECTURE

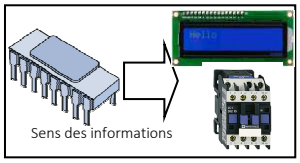
Le processeur effectue une lecture si on y rentre des informations (au clavier ou avec un capteur, …) pour qu'elles soient utilisées par le programme. *Titi ← LIRECLAVIER*



ECRITURE

Le processeur effectue une écriture si celui-ci envoie une information (sur un écran, un préactionneur, …)

ECRIREECRAN (Toto)



Les tests.

Il n'y a que deux formes possibles pour un test ; la première est la plus simple, la seconde la plus complexe.

```
SI <condition> ALORS
    action_alors
FINSI
```

```
SI <condition> ALORS
    action_alors
SINON
    Action_sinon
FINSI
```

```
SELONQUE
<condition_1> FAIRE action_1
…
<condition_n> FAIRE action_n
FINSELONQUE
```

Une **condition** est une **expression** dont la valeur est VRAIE ou FAUSSE. Cela peut donc être :

- une **variable** (ou une expression) de type booléen ;
- une **comparaison** réalisée à partir des opérateurs : =, ≠, >, <, ≥, ≤).

Les boucles.

On les appelle aussi structures itératives ou structures répétitives.

Boucle Tant que

```
TantQue <condition vraie>
    actions
FinTantQue
```

Boucle Répéter jusqu'à

```
Répeter
    actions
Jusqu'à <condition vraie>
```

Boucle Pour - Compter en bouclant

```
Pour compteur ← val_inf à val_sup Pas val_pas
    actions_à_répéter
Compteur suivant
```

4 - METHODE DE TRAVAIL

Pour faciliter le travail d'écriture et de débogage il faut,

1- Décomposer le problème en sous problèmes ;
2- Répertorier les variables et leur type utiles pour chaque sous problème ;
3- Créer un algorithme pour chaque sous problème (mettre des commentaires dans les lignes de code + prendre des noms de variables logiques pour tout le monde) ;
4- Tester le fonctionnement de chaque algorithme de sous problème ;
5- Créer un algorithme général gérant les sous algorithmes ;
6- Tester le fonctionnement général ;
7- Traduire chaque sous problème dans le langage de programmation désiré ;
8- Tester chaque sous-programme ;
9- Traduire l'algorithme général dans le langage de programmation ;
10- Tester le fonctionnement général.