

LANGAGE ALGORITHMIQUE

1 Introduction

Alors qu'ils n'existaient pas il y a 50 ans, les ordinateurs ont aujourd'hui envahi notre environnement. Bien que très puissant par la quantité d'informations qu'il peut engranger et le grand nombre d'opérations qu'il peut exécuter par seconde, un ordinateur n'est qu'une machine capable d'exécuter automatiquement une série d'opérations simples qu'on lui a demandé de faire.

Pour résoudre un problème à l'aide d'un ordinateur, il faut:

1. analyser ce problème: définir avec précision les résultats à obtenir, les informations dont on dispose, ...
2. déterminer les méthodes de résolution: il s'agit de déterminer la suite des opérations à effectuer pour obtenir à partir des données la solution au problème posé. Cette suite d'opérations constitue un **algorithme**. Parmi tous les algorithmes fournissant la solution, il faudra choisir le plus efficace.
3. formuler l'algorithme définitif: cette étape doit faciliter la résolution sur ordinateur par l'expression de l'algorithme dans un formalisme adéquat (langage de description d'algorithme: LDA, organigramme, arbre programmatique, ...).
4. traduire l'algorithme dans un langage de programmation adapté.

L'Encyclopédia Universalis donne la définition suivante de l'Algorithme :

« Un algorithme est une suite de règles à appliquer dans un ordre déterminé à un nombre fini de données pour arriver, en un nombre fini d'étapes, à un certain résultat, et cela indépendamment des données. »

Le mot algorithme provient du nom d'un célèbre mathématicien arabe de la première moitié du IX^e siècle : MUHAMMAD IBN MUSA AL KHWARISMI.

Les algorithmes sont fondamentaux au sens où ils sont indépendants à la fois de l'ordinateur qui les exécute et des langages dans lequel ils sont traduits.

2 Qualités d'un algorithme

Pour obtenir un bon programme il faut partir d'un bon algorithme qui doit posséder entre autres, les qualités suivantes :

- Etre clair, facile à comprendre par tous ceux qui le lisent.
- Etre le plus général possible pour répondre au plus grand nombre de cas possibles.
- Etre d'une utilisation aisée même par ceux qui ne l'ont pas écrit (Messages pour l'introduction des données...)
- Etre conçu de manière à limiter le nombre d'opérations à effectuer et la place occupée en mémoire.

Une des meilleures façons de rendre un algorithme clair et compréhensible est d'utiliser un langage de description structuré n'utilisant qu'un petit nombre de structures indépendantes du langage de programmation utilisé.

3 Éléments représentatifs du langage algorithmique

3.1 Les MOTS

On distingue trois familles de mots :

- Les mots CLES
- Les mots INSTRUCTIONS
- Les mots DELIMITEURS

3.1.1 MOTS CLES

Les mots clés définissent la structure algorithmique utilisée

Exemples de mots clés :

- *SI ... ALORS... SINON...* : définissent une structure *ALTERNATIVE*
- *REPETER... JUSQU'A...* : définissent une structure *ITERATIVE*

Un mot clé est toujours suivi :

- Soit d'une expression conditionnelle écrite entre guillemets
- Soit d'un ou plusieurs mots instructions

Exemple :

```
- SI « condition 1 »  
  - ALORS  
    - mot instruction 1  
    - mot instruction 2  
  - SINON  
    - REPETER  
      - mot instruction 3  
      - mot instruction 4  
    - JUSQU'A « condition 2 »  
- FIN SI
```

3.1.2 MOTS INSTRUCTIONS

Se sont des verbes d'action qui caractérisent la nature des opérations à effectuer sur une ou plusieurs données.

Un mot instruction est toujours suivi entre guillemets :

- De la désignation de l'objet sur lequel il s'applique .

Exemple : - LIRE « Capteur S1 »

- Eventuellement de la description de l'opération à appliquer à l'objet.

Exemple : - FAIRE « Compteur = Compteur – 1 »

3.1.3 MOTS DELIMITEURS

Les mots délimiteurs fixent :

- Les bornes d'ENTRÉE et de SORTIE de l'algorithme
- Les bornes d'ENTRÉE et de SORTIE des différentes structures utilisées dans l'algorithme si ces bornes ne sont pas définies par la structure elle même.

DEBUT et FIN sont les seuls mots délimiteurs et peuvent être suivi éventuellement d'un mot clé.

Exemple : FIN SI

4 Représentation des algorithmes

4.1 Représentation LITTERALE

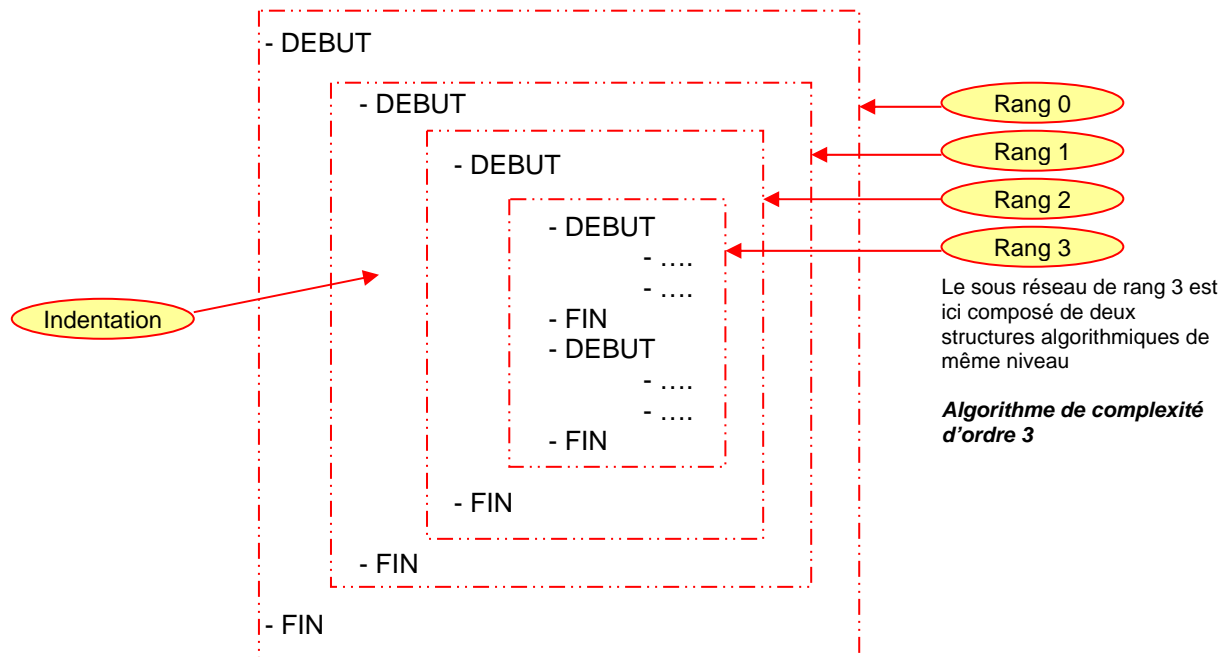
Aux règles de représentation littérales précédemment définies nous ajouterons les règles d'écriture suivantes afin de faciliter la lecture de l'algorithme :

L'écriture sera **indentée**¹ afin de faire apparaître l'algorithme comme un réseau principal comportant une borne d'entrée et une borne de sortie et chacune des structures qui le constitue comme un sous réseau présentant aussi une entrée et une sortie.

Règles :

- Le dernier sous réseau ouvert doit être le premier fermé
- Le nombre de fermetures doit être égal au nombre d'ouvertures

Le rang de la dernière paire de mots délimiteurs représente le degré de complexité de l'algorithme.



4.2 Représentation GRAPHIQUE ou ALGORIGRAMME

La représentation graphique permet une lecture aisée des algorithmes mais présente toutefois l'inconvénient de consommer une place importante. Elle utilise les symboles de la norme NF Z 67-010 dont les principaux sont les suivants :

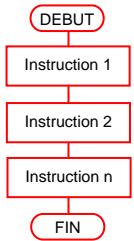


	Début, Fin Début, fin ou interruption d'un organigramme		Embranchement Pour représenter une décision ou un aiguillage après un test
	Renvoi Utilisé deux fois pour assurer la continuité d'un organigramme scindé		Traitement Opération ou groupe d'opérations sur des données
	Commentaire Pour donner des indications marginales		Entrée / Sortie Mise à disposition d'une information ou enregistrement d'une information
	Liaison Le sens général des liaisons doit être : - de haut en bas - de gauche à droite		Procédé prédéfini Portion de programme considérée comme une simple opération

¹ Indenter : Tabuler le texte en insérant des espaces au début de chaque nouvelle structure afin de la mettre en évidence.

5 Structures algorithmiques fondamentales

5.1 Structure LINEAIRE ou SEQUENTIELLE

C'est une SUITE D'ACTIONs à exécuter successivement dans l'ordre de leur énoncé

Algorithme LITTERAL	Algorithme GRAPHIQUE	Structure GRAFCET
- Instruction 1 - Instruction 2 - ... - Instruction n		<p><i>Séquence</i></p> 
Les mots instructions sont écrits au même rang sur des lignes successives.		
EXEMPLE		
- DEBUT - FAIRE "AC=1" - FAIRE "EV=1" - OUVRIR "Vanne 2" - FIN		

5.2 Structures ITERATIVES

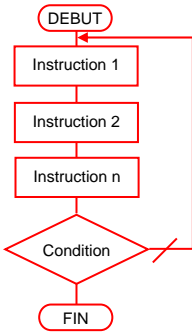
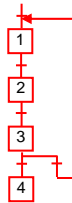
Par ITERATION on désigne toute répétition de l'exécution d'un traitement. Trois types de structures itératives sont à distinguer:

- La structure REPETER....JUSQU'A...
- La structure REPETER... TANT QUE...
- La structure POUR... A... REPETER...

Dans les deux premier cas, le nombre de répétitions n'est pas connu à l'avance et dépend d'un ou plusieurs évènements extérieurs.

Dans le dernier cas, le nombre de répétitions est connu à l'avance, il est CONSIGNE.

5.2.1 Structure " REPETER... JUSQU'A..."

Algorithme LITTERAL	Algorithme GRAPHIQUE	Structure GRAFCET
- REPETER - Instruction 1 - Instruction 2 - Instruction n - JUSQU'A "Condition vraie"		 <p><i>Reprise de séquence</i></p>
En anglais: REPEAT... UNTIL...		
Le traitement est exécuté une première fois dès l'entrée dans la structure, il se répète jusqu'à ce que la condition soit vérifiée.		

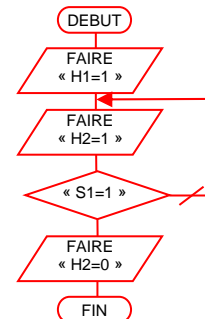
EXEMPLE

A l'apparition d'un défaut on provoque simultanément la mise sous tension :

- D'un signal lumineux H1
- D'un avertisseur sonore H2

L'arrêt de l'avertisseur sonore se fait lorsque l'opérateur acquitte le défaut par une action sur le bouton poussoir S1

- DEBUT
 - FAIRE "H1=1"
 - **REPETER**
 - FAIRE "H2=1"
 - **JUSQU'A** « S1=1 »
 - FAIRE « H2=0 »
 - FIN



5.2.2 Structure "REPETER... TANT QUE..."

Algorithme LITTÉRAL	Algorithme GRAPHIQUE	Structure GRAFCET
<p>- REPETER</p> <ul style="list-style-type: none"> - Instruction 1 - Instruction 2 - Instruction n <p>- TANT QUE "Condition vraie"</p>	<pre> graph TD DEBUT([DEBUT]) --> I1[Instruction 1] I1 --> I2[Instruction 2] I2 --> In[Instruction n] In --> C{Condition} C -- Vraie --> I1 C -- Fausse --> FIN([FIN]) </pre>	<p style="text-align: center;"><i>Reprise de séquence</i></p>

En anglais: REPEAT... WHILE...

Le traitement est exécuté une première fois dès l'entrée dans la structure, il se répète tant que la condition est vérifiée. Les deux structures itératives « REPETER... JUSQU'A... » et « REPETER... TANT QUE... » sont strictement équivalentes

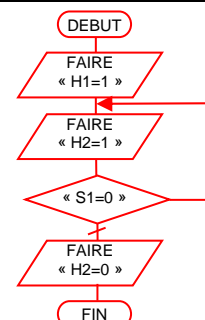
EXEMPLE (même exemple que précédemment)

A l'apparition d'un défaut on provoque simultanément la mise sous tension :

- D'un signal lumineux H1
- D'un avertisseur sonore H2

L'arrêt de l'avertisseur sonore se fait lorsque l'opérateur acquitte le défaut par une action sur le bouton poussoir S1

- DEBUT
 - FAIRE "H1=1"
 - **REPETER**
 - FAIRE "H2=1"
 - **TANT QUE** « S1=0 »
 - FAIRE « H2=0 »
 - FIN



5.2.3 Structure " POUR... A... REPETER..."

Algorithme LITTERAL	Algorithme GRAPHIQUE	Structure GRAFCET
<p>- POUR V=Vi A V=Vf Pas=P</p> <p>- REPETER</p> <ul style="list-style-type: none"> - Instruction 1 - Instruction 2 - Instruction n <p>- FIN POUR</p>		<p><i>Reprise de séquence</i></p>
<p>En anglais: FOR... TO... STEP... NEXT...</p> <p>Dans cette structure la sortie de la boucle d'itération s'effectue lorsque le nombre de répétitions est atteint.</p> <p>D'où l'emploi d'une variable V dite de contrôle d'itération et définie par :</p> <ul style="list-style-type: none"> • sa VALEUR INITIALE : Vi • sa VALEUR FINALE : Vf • son PAS DE VARIATION : P <p>Cette structure s'utilise si le nombre NR de répétitions est connu :</p> $NR = \frac{ V_f - V_i }{ P }$ <p>Exemple : Pour Vf=5 ; Vi=0 ; P= +1 : NR=5 Pour Vf=0 ; Vi=8 ; P= -2 : NR=4</p> <p style="text-align: center;">EXEMPLE</p> <p>Le guichetier d'une salle de cinéma tape sur son clavier le nombre de billets demandé par le client. Le distributeur déroule alors les billets en sectionne la bande après le dernier billet :</p>		
<p>- DEBUT</p> <p>- TAPER "Nombre de billets NB"</p> <p>- POUR V=0 A V=NB</p> <p>- REPETER</p> <ul style="list-style-type: none"> - DELIVRER "1 billet" <p>- FIN POUR</p> <p>- COUPER « Bande »</p> <p>- FIN</p>		

5.3 Structures ALTERNATIVES

Ces structures désignent toute situation n'offrant que deux issues possibles s'excluant mutuellement. Il existe deux types de structures alternatives :

- La structure alternative COMPLETE
- La structure alternative REDUITE

5.3.1 Structure " ALTERNATIVE COMPLETE "

Algorithme LITTERAL	Algorithme GRAPHIQUE	Structure GRAFCET
- SI « CONDITION VRAIE » - ALORS - Traitement 1 - SINON - Traitement 2 - FIN SI		Divergence en OU ou Sélection de séquence
En anglais : IF... THEN... ELSE... L'exécution d'un des deux traitements dépend du résultat d'un test : <ul style="list-style-type: none"> • Si le test est VRAI le premier traitement est exécuté. • Si le test est FAUX c'est le deuxième traitement qui s'effectue. 		
EXEMPLE Sur une chaîne de conditionnement un dispositif de tri permet de diriger les caisses de Masse supérieure ou égale à 20 Kg vers le tapis 1 et les autres vers le tapis 2 en comptabilisant le nombre de caisses.		
- DEBUT - PESER "Caisse" - SI "P>=20Kg" - ALORS - AIGUILLER « caisse vers tapis 1 » - FAIRE « C1=C1+1 » - SINON - AIGUILLER « caisse vers tapis 2 » - FAIRE « C2=C2+1 » - FIN SI - FIN		

5.3.2 Structure " ALTERNATIVE REDUITE "

Algorithme LITTERAL	Algorithme GRAPHIQUE	Structure GRAFCET
<p>- SI « CONDITION VRAIE »</p> <p>- ALORS</p> <p>- Traitement 1</p> <p>- FIN SI</p>	<pre> graph TD DEBUT([DEBUT]) --> Condition{Condition} Condition -- Vrai --> Traitement1[Traitement 1] Traitement1 --> FIN([FIN]) Condition -- Faux --> Join(()) Traitement1 --> Join Join --> FIN </pre>	<p>Saut d'étapes</p> <pre> graph TD 1[1] --> 2[2] 2 -- / --> 4[4] 4 --> 3[3] </pre>
<p>En anglais : IF... THEN...</p> <p>Seule la situation correspondant à la validation de la condition entraîne l'exécution du traitement dans le cas où la condition n'est pas satisfaite, le traitement n'est pas exécuté et la structure est abandonnée.</p>		
<p>EXEMPLE</p> <p>Sur une chaîne de conditionnement un dispositif de tri permet de diriger les caisses de Masse supérieure ou égale à 20 Kg vers le tapis 1, les autres continuent d'avancer sur le même tapis 2.</p>		
<p>- DEBUT</p> <p>- PESER "Caisse"</p> <p>- SI "P>=20Kg"</p> <p>- ALORS</p> <p>- AIGUILLER « caisse vers tapis 1 »</p> <p>- FIN SI</p> <p>- FIN</p>	<pre> graph TD DEBUT([DEBUT]) --> PESER[PESER « caisse »] PESER --> P20{P>=20Kg} P20 -- Vrai --> AIGUILLER[AIGUILLER « Vers tapis 1 »] AIGUILLER --> FIN([FIN]) P20 -- Faux --> Join(()) AIGUILLER --> Join Join --> FIN </pre>	

6 Structures emboîtées

Le niveau de difficultés des problèmes posés par l'automatisation des processus industriels conduisent souvent à rédiger des algorithmes de degré de complexité d'ordre supérieur à 1 utilisant les structures fondamentales en combinaisons emboîtées.

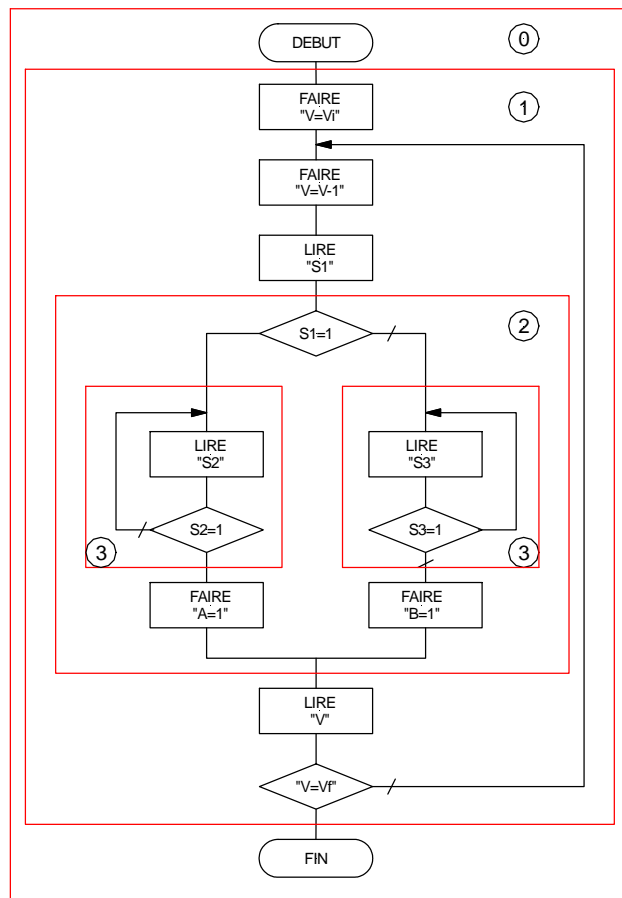
L'algorithme suivant intègre dans un ordre d'emboîtement décroissant :

- Une structure itérative POUR... A... REPETER...
- Une structure alternative COMPLETE
- Deux structures itératives REPETER... JUSQU'A... et REPETER... TANT QUE...

Son degré de complexité est d'ordre 3


```

- DEBUT
- POUR V=Vi A V= Vf
- REPETER
  - FAIRE « V=V-1 »
  - LIRE « S1 »
  - SI « S1=1 »
    - ALORS
      - REPETER
        - LIRE « S2 »
        - JUSQU'A « S2=1 »
        - FAIRE « A=1 »
      - SINON
        - REPETER
          - LIRE « S3 »
          - TANT QUE « S3=1 »
          - FAIRE « B=1 »
    - FIN SI
  - FIN POUR
- FIN
    
```



1	Introduction	1
2	Qualités d'un algorithme	1
3	Éléments représentatifs du langage algorithmique	2
3.1	Les MOTS	2
3.1.1	MOTS CLES	2
3.1.2	MOTS INSTRUCTIONS	2
3.1.3	MOTS DELIMITEURS	2
4	Représentation des algorithmes	3
4.1	Représentation LITTERALE	3
4.2	Représentation GRAPHIQUE ou ALGORIGRAMME	3
5	Structures algorithmiques fondamentales	4
5.1	Structure LINEAIRE ou SEQUENTIELLE	4
5.2	Structures ITERATIVES	4
5.2.1	Structure " REPETER... JUSQU'A..."	4
5.2.2	Structure " REPETER... TANT QUE..."	5
5.2.3	Structure " POUR... A... REPETER..."	6
5.3	Structures ALTERNATIVES	7
5.3.1	Structure " ALTERNATIVE COMPLETE"	7
5.3.2	Structure " ALTERNATIVE REDUITE"	8
6	Structures emboîtées	8