01/10/2015

COURS D'ALGORITHME.

Initiation générale.

669 35 43 10 / 655 34 42 38 / 624 05 56 40

massaleidamagoe@yahoo.fr

massaleidamagoe2015@gmail.com

www.massaleidamagoe2015.net

Mazoughou Goépogui

MAGOE TECHNOLOGIE

I. Définitions.

I.1. Définition d'un algorithme.

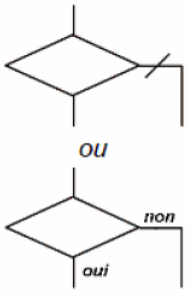
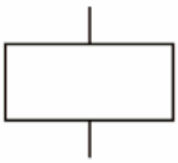
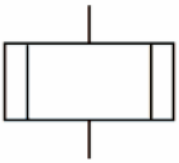
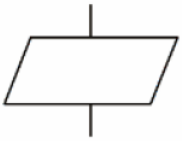
Un algorithme est l'ensemble des règles opératoires ordonnant à un processeur d'exécuter dans un ordre déterminé un nombre d'opérations élémentaires. Il impose une programmation de type structurée.

I.2. Définition algorigramme.

Algorigramme est une représentation graphique de l'algorithme utilisant des symboles normalisés. En réalité c'est un diagramme qui permet de représenter et d'étudier le fonctionnement des automatismes de types séquentiels comme les chronogrammes ou le GRAFCET mais davantage réservé à la programmation des systèmes microinformatiques ainsi qu'à la maintenance.

Le diagramme est une suite de directives composées d'actions et de décisions qui doivent être exécutés selon un enchaînement strict pour réaliser une tâche (ou séquence).

Les principaux symboles utilisés sont données ci-dessous.

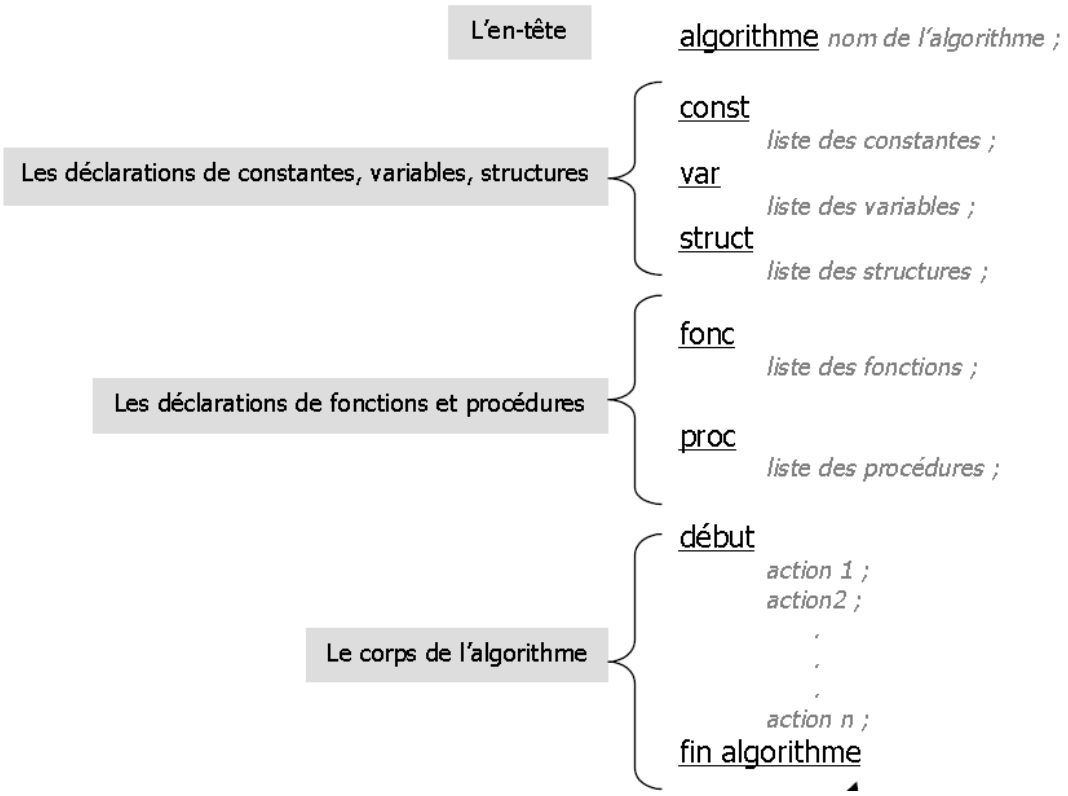
SYMBOLE	DÉSIGNATION	SYMBOLE	DÉSIGNATION
	**début** ou **fin** d'un algorithme		**Test ou Branchement conditionnel** décision d'un choix parmi d'autres en fonction des conditions
	symbole général de **« traitement »** opération sur des données, instructions, … ou opération pour laquelle il n'existe aucun symbole normalisé		**sous-programme** appel d'un sous-programme
	**entrée / sortie**		**Liaison** Les différents symboles sont reliés entre eux par des lignes de liaison. Le cheminement va de haut en bas et de gauche à droite. Un cheminement différent est indiqué à l'aide d'une flèche
	**commentaire**		

Remarques.

- Les symboles de début et de fin de programme ne sont pas toujours représentés.

II. Structure d'un algorithme.

La structure générale d'un algorithme est donnée ci-dessous.



1. **L'entête.** Il permet tout simplement d'identifier l'algorithme.
2. **Les déclarations.** C'est une liste exhaustive d'objets, de grandeurs utilisés et manipulés dans le corps de l'algorithme. Cette liste est placée en début d'algorithme.
3. **Le corps.** C'est dans cette de l'algorithme que placées les tâches (instructions) à exécuter.
4. **Les commentaires.** Ils permettent une interprétation aisée de l'algorithme. L'utilisation de commentaires est vivement conseillée.

III. Les structures algorithmiques fondamentales.

Les opérations élémentaires relatives à la résolution d'un problème peuvent, en fonction de leur enchainement, être organisées suivant quatre familles de structures algorithmiques fondamentales.

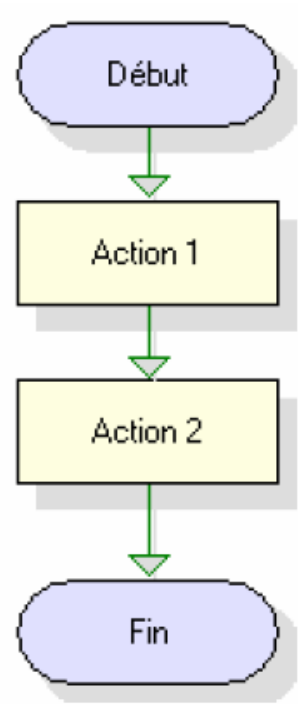
1. Structures linéaires.
2. Structures alternatives.
3. Structures de choix.
4. Structure itératives (ou répétitives).

III.1. Structure linéaire.

La structure linéaire se caractérise par une suite d'actions à exécuter successivement dans l'ordre énoncé.

Notation :

Début
Action 1
Action 2
Fin



Exemple en langage C.
{ Action 1 ; }
{ Action 2 ; }

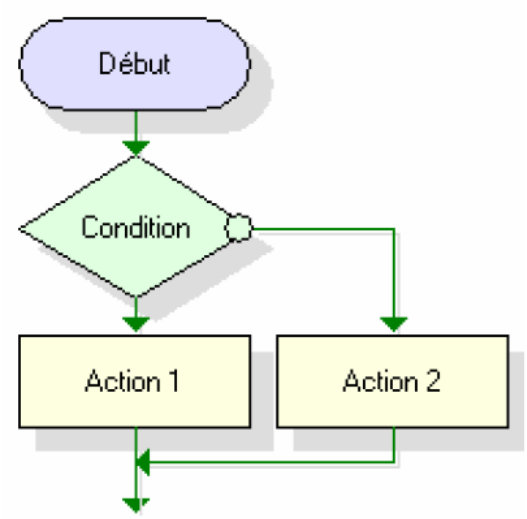
III.2. Structure alternative.

Cette structure offre le choix entre deux séquences s'excluant mutuellement. On peut rencontrer deux types de structures alternatives : la structure alternative complète et la structure alternative simple.

a) Structure alternative complète.

Notation :

Début
Si Condition
Alors Action 1
Sinon Action 2

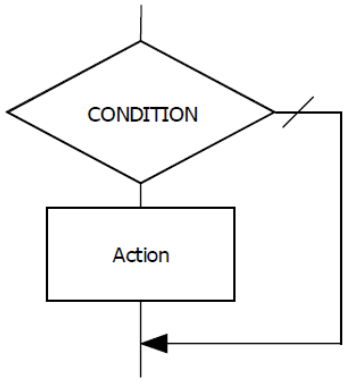


Exemple en langage C.
If (Condition)
{ Action 1 ; }
Else
{ Action 2 ; }

b) Structure alternative réduite.

Notation :

Début
Si Condition
Alors Action



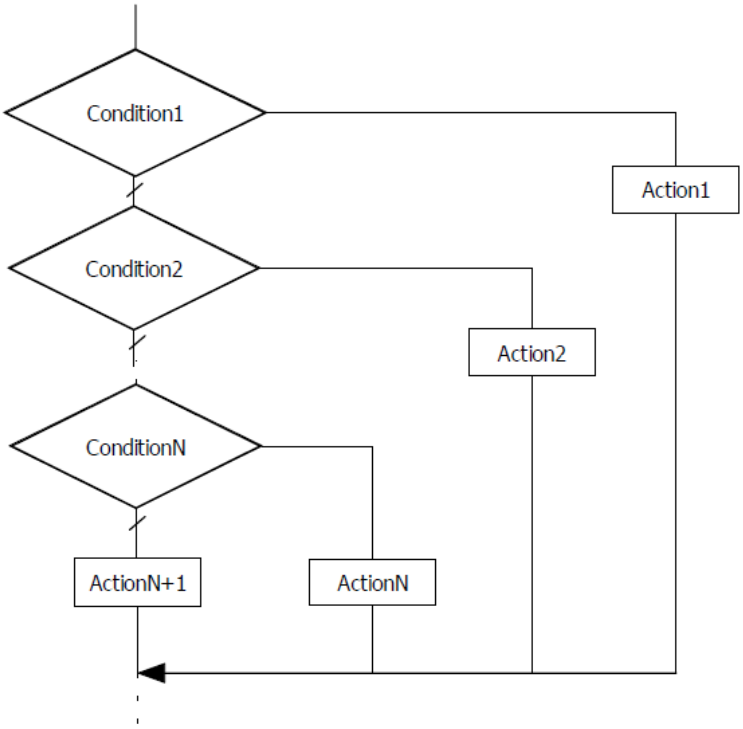
Exemple en langage C.
If (Condition)
{ Action ; }

III.3. Structure de choix.

La structure de choix permet, en fonction de plusieurs conditions de type booléen, d'effectuer des actions différentes suivant les valeurs que peut prendre une même variable.

Notation :

suivant valeur **faire**
valeur 1 : action 1
valeur 2 : action 2
.
.
.
valeur N : action N
sinon action N+1



Exemple en langage C.

switch (valeur)
{
case valeur 1: action 1;
break;
case valeur 2: action 2;
break;
.
.
.
case valeur N: action N;
break;
default : action N+1
}

III.4. Structure itérative (ou répétitive).

Cette structure répète l'exécution d'une opération ou d'un traitement. Deux cas peuvent arriver.

III.4.1. Le nombre de répétition n'est pas connu ou est variable.

Là également deux cas peuvent arriver.

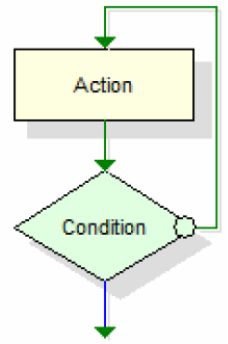
a) Structure « *répéter jusqu'à* ».

Dans cette structure le traitement est exécuté une première fois puis sa répétition se poursuit jusqu'à ce que la condition soit vérifiée.

Notation :

répéter
action ;
jusqu'à condition vraie ;

Remarque. En faisant de sorte que la condition soit toujours vraie, l'action se répétera de façon infinie : c'est la boucle infinie.



Exemple en langage C.

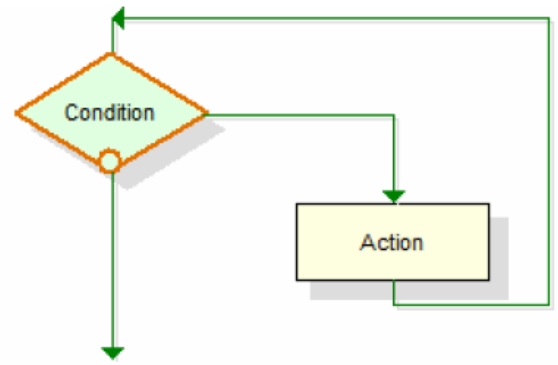
action ;
while condition ;

b) Structure *« tant que … faire »*.

Dans cette structure, on commence par tester la condition ; si elle vraie, le traitement est exécuté.

Notation :

tant que condition **faire**
action ;
fin tant que ;



Exemple en langage C.
while (condition)
action ;

III.4.2. Le nombre de répétition est connu.

Dans cette structure, la sortie de la boucle d'itération s'effectue lorsque le nombre de répétition souhaité est atteint.

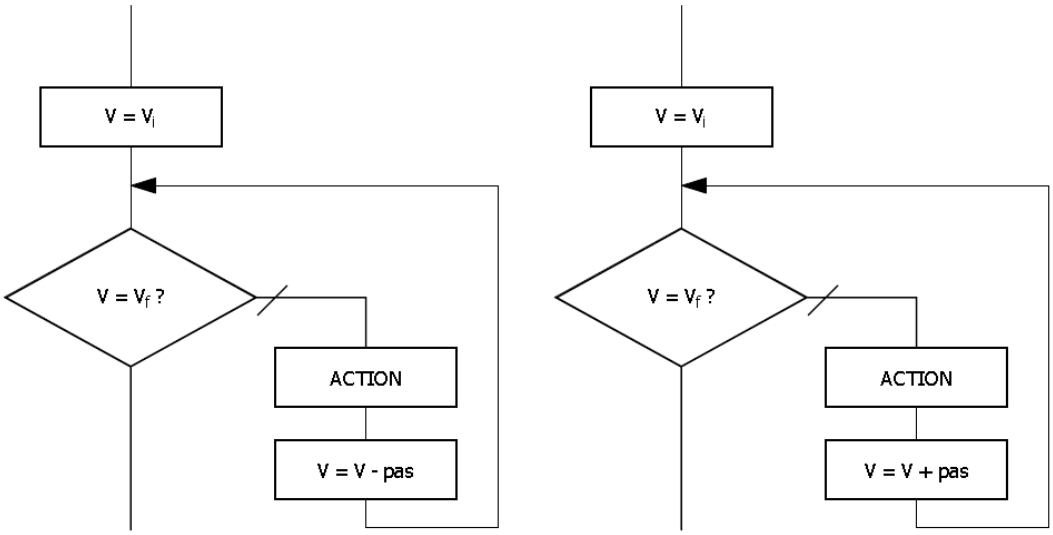
On utilise donc une variable (ou indice) de contrôle d'itération caractérisée par :

- Sa valeur initiale ;
- Sa valeur finale ;
- Son pas de variation.

Si la valeur finale de l'indice est inférieure à sa valeur initiale, le pas de variation est négatif, la structure est dite « ***pour décroissante*** » ; dans le cas contraire, le pas est positif et la structures est dite « ***pour croissante*** »

Notation :

pour variable **de** début **à** fin **pas** n **faire**
action ;
fin pour ;



V : variable ;
V_i : valeur initiale de V ;
V_f : valeur finale de V ;

Exemple en langage C.

for ([expression_1] ; [expression_2] ; [expression_3])
action ;

Les crochets signifient que leur contenu est facultatif. Lorsque *expression_2* est absente, elle est considérée comme vraie. D'une manière générale, en C, la structure **for** peut être remplacée par **while** comme indiqué ci-dessous.

expression_1 ;
***while** (expression_2)*
{instruction ;
expression_3 ;}