



ESPRIT

Guide des ORGANIGRAMMES

TABLE DES MATIERES

1. Introduction.....	3
2. Définitions.....	3
a) Algorithme	3
b) Organigramme.....	3
3. La norme ISO 5807.....	4
a) Symboles	4
b) Exemple.....	5
4. Les structures alternatives.....	6
a) SI... ALORS...[IF... THEN...]	6
b) SI... ALORS... SINON...[IF... THEN... ELSE...].....	7
5. Les structures itératives ou répétitives	8
a) TANTQUE... FAIRE...[WHILE... DO...]	8
b) FAIRE... JUSQU'A...[DO... UNTIL...]	9
c) POUR... FAIRE...[FOR... DO...]	10
6. Exercices	11
a) Exercice 1	11
b) Exercice 2	12

1. Introduction

Avant de se lancer dans la programmation, il serait intéressant d'avoir une idée claire sur la structure du programme qu'on va développer.

Pour cela, nous pouvons représenter le programme avec un algorithme ou un organigramme.

Le premier se présente dans une structure linéaire comme un programme et le deuxième permet d'avoir une représentation graphique qui facilite encore mieux la compréhension.

Faire un organigramme peut s'avérer très utile par exemple dans le travail en groupe. Dans ces conditions plusieurs intervenants participent à la programmation et la modification du code. Ces modifications peuvent être représentées plus rapidement et facilement à l'aide d'organigrammes. Les organigrammes permettent même d'éclairer le concepteur lui-même sur des idées qu'il avait eu. La réalisation d'un organigramme est tout aussi importante que de mettre des commentaires dans le programme.

Il est donc important de savoir cette méthodologie de représentation.

2. Définitions

a) Algorithme

Un algorithme est un énoncé d'une suite d'opérations permettant de donner la réponse à un problème.


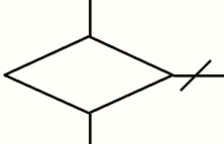
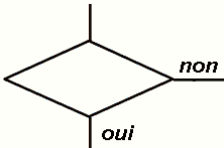
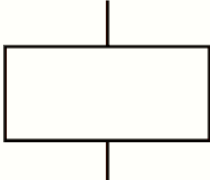
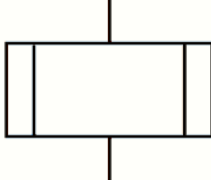
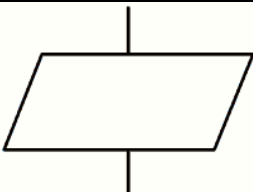



b) Organigramme

C'est une représentation graphique de l'algorithme. Pour le construire, on utilise des symboles normalisés.

3. La norme ISO 5807

Les principaux symboles rencontrés dans un algorithme sont représentés dans le tableau ci-dessous. L'algorithme permet une vision globale mais reste limité aux études peu complexes.

a) Symboles

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

b) Exemple

Cet algorithme saisit une valeur entière et affiche son double si cette donnée est inférieure à un seuil donné.

Algorithme	Organigramme
<p>Algorithme SimpleOuDouble</p> <p>constante (SEUIL : entier) $\leftarrow 10$</p> <p>variable val : entier</p> <p>début</p> <p> afficher("Donnez-moi un entier : ")</p> <p> saisir(val)</p> <p> si val < SEUIL</p> <p> alors val \leftarrow val \times 2</p> <p> fsi</p> <p> afficher("Voici la valeur finale : ", val)</p> <p>fin</p>	<pre> graph TD Debut([Début]) --> Afficher1[Afficher ("Donner moi un entier ")] Afficher1 --> Saisir[/Saisir (val)/] Saisir --> Decision{Val < seuil} Decision -- oui --> Calcul[Val ← val x 2] Decision -- non --> Afficher2[Afficher ("Voici la valeur finale : ", val)] Calcul --> Afficher2 Afficher2 --> Fin([Fin]) </pre>
Programme C	
<pre> #include <stdio.h> #include <conio.h> #define SEUIL 10 void main() { int val; printf ("Donnez moi un entier : "); scanf ("%d",&val); if (val < SEUIL) { val = val * 2; } printf("Voici la valeur finale : %d",val); getch();} </pre>	

4. Les structures alternatives

a) SI... ALORS...[IF... THEN...]

Si la condition est vraie alors l'action est exécutée, sinon elle ne l'est pas.

Organigramme	Algorithme
<pre> graph TD Start(()) -- si --> Cond{condition} Cond -- alors --> Action1[action1] Action1 --> Join(()) Cond -- sinon --> Join </pre>	SI (condition vraie) ALORS BLOC 1 D'INSTRUCTIONS FIN SI
	Programme C If (condition) { BLOC 1 D'INSTRUCTIONS }

b) SI... ALORS... SINON...[IF... THEN... ELSE...]

Exécution d'un test, selon le résultat, le programme se poursuit vers une branche ou vers l'autre.

Si la condition est vraie alors l'action1 est exécutée, sinon c'est l'action 2 qui sera exécutée.

Organigramme	Algorithme
<pre> graph TD Start(()) -- si --> Condition{condition} Condition -- alors --> Action1[action1] Condition -- sinon --> Action2[action2] Action1 --> Join(()) Action2 --> Join Join -- Finsi --> End(()) </pre>	SI (condition vraie) ALORS BLOC 1 D'INSTRUCTIONS SINON BLOC 2 D'INSTRUCTIONS FINSI
	Programme C
	If (condition) { BLOC 1 D'INSTRUCTIONS } else { BLOC 2 D'INSTRUCTIONS }

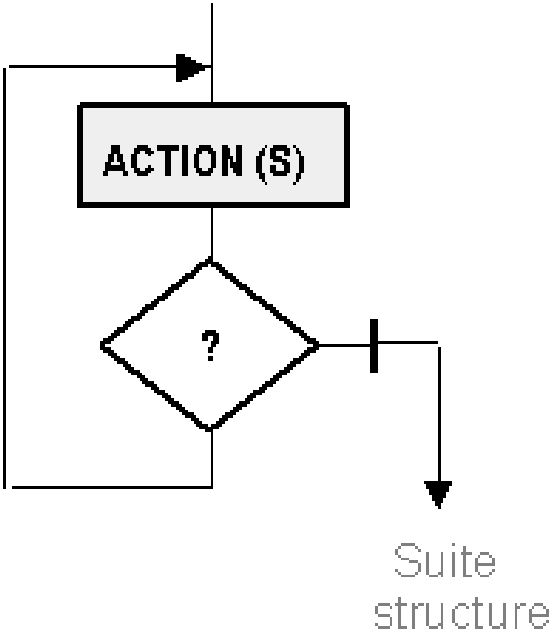
5. Les structures itératives ou répétitives

a) TANT QUE... FAIRE...[WHILE... DO...]

Organigramme	Algorithme
<pre> graph TD Entry(()) --> Decision{?} Decision --> Action[ACTION(S)] Action --> Entry Decision --> Exit(()) Exit --- Suite[Suite structure] </pre>	TANT QUE (condition est vraie) FAIRE BLOC D'INSTRUCTIONS FIN TANT QUE
	Programme C while (condition) { BLOC D'INSTRUCTIONS }

b) repeter... tant que... [DO... While...]

Contrairement à la boucle tant que, l'action est ici exécutée au moins une fois.

Organigramme	Algorithme
	<p>REPETER</p> <p>BLOC D'INSTRUCTIONS</p> <p>TANT QUE (condition est vraie)</p>
	<p>Programme C</p> <pre>do { BLOC D'INSTRUCTIONS } While (condition) ;</pre>

c) POUR... FAIRE...[FOR... DO...]

Cette boucle est utilisée quand le nombre de boucles à exécuter est connu avant.

Organigramme	Algorithme
<pre> graph TD Start(()) --> Init[i ← <valeur_initiale>] Init --> Actions[actions] Actions --> Inc[i ← i + n] Inc --> Decision{i ≥ <valeur_finale>} Decision -- faux --> Actions Decision -- vraie --> Exit(()) </pre>	POUR i de valeur_initiale à valeur _finale pas n { BLOC D'INSTRUCTIONS }
	Programme C
	<pre> int i ; for (i=valeur_initiale ;i<=valeur_finale ;i=i+n) { BLOC D'INSTRUCTIONS } </pre>

6. Exercices

a) Exercise 1

Saisir des valeurs, les traiter, et s'arrêter à la saisie de la valeur d'arrêt -1 ou après avoir saisi 5 données.

Organigramme	Algorithme
	Constantes (STOP : entier) $\leftarrow -1$ (MAX : entier) $\leftarrow 5$ Variables nbVal, val : entiers Début nbVal $\leftarrow 0$ tant que val \neq STOP et nbVal < MAX faire nbVal \leftarrow nbVal + 1 ... {traitement de la valeur saisie} saisir(val) ftq afficher(val, nbVal)

b) Exercise 2

Ecrire le programme C correspondant à cet organigramme.

Organigramme	Programme C
<pre>graph TD; Debut([Début]) --> Input[/A, B/]; Input --> Decision{A > B}; Decision -- Vrai --> Process1[C ← A - B]; Decision -- Faux --> Process2[C ← B - A]; Process1 --> Merge(()); Process2 --> Merge; Merge --> Output[/C/]; Output --> Fin([Fin]);</pre>	