La programmation

Synthèse

I. Algorithme et algorigramme

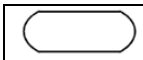
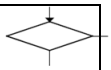
Algorithme : c'est un ensemble de règles opératoires rigoureuses, ordonnant à un processeur d'exécuter, dans un ordre déterminé, un nombre fini d'opérations élémentaires appelées « instructions ».

Organisation d'un algorithme

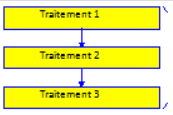
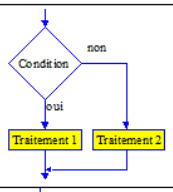
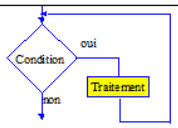
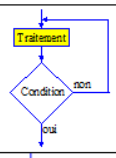
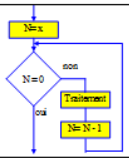
L'en-tête : Dans cette partie le concepteur donne un **nom** à l'algorithme. Il définit le traitement effectué et les données auxquelles il se rapporte.
La partie déclarative : Dans cette partie, le concepteur décrit les différents « **objets** » que l'algorithme utilise.

Les constantes	Les variables
Ce sont des « objets » constants dans tout l'algorithme. Déclaration : nom_constante=valeur *Exemple :* Pi = 3,1416 La déclaration de constantes symboliques permet de donner un nom à un objet constant dans tout l'algorithme et ensuite de faire référence à cet objet par son nom plutôt que par sa valeur.	Ce sont des « objets » dont la valeur peut changer au cours de l'exécution de l'algorithme. Déclaration : nom_variable :type Le « type » peut être : nombres entiers, octets, chaines de caractère,…

La partie exécutive
Elle est délimitée par les mots « début » et « fin ».
Algorigramme : c'est une représentation graphique de l'algorithme. Pour le construire, on utilise des symboles normalisés. Ci-dessous, quelques exemples :

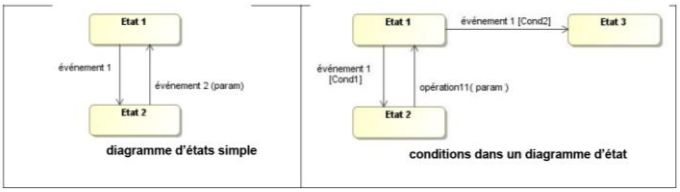
				
Début / fin	traitement	Sous-programme (macro)	Entrée / sortie	Test / condition

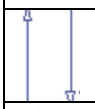
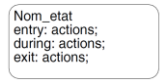
Exemples de structures :

Structure	Algorigramme	Algorithme
Structure linéaire La structure linéaire se caractérise par une suite d'actions à exécuter successivement dans l'ordre de leur énoncé.		FAIRE « traitement 1 » FAIRE « traitement 2 » FAIRE « traitement 3 »
Structure alternative ou conditionnelle Une structure alternative n'offre que deux issues possibles s'excluant mutuellement. Les structures alternatives définissent une fonction de choix ou de sélection entre l'exécution de l'un ou de l'autre des deux traitements.		SI « condition » vraie ALORS FAIRE « traitement 1 » SINON FAIRE « traitement 2 » FIN SI
Structure répétitive ou itérative (boucle avec pré-test) Dans cette structure on commence par tester la condition, si elle est vraie alors le traitement est exécuté.		TANT QUE « condition » vraie FAIRE « traitement » FIN TANT QUE
Structure répétitive ou itérative (boucle avec post-test) Dans cette structure le traitement est exécuté une première fois puis sa répétition se poursuit jusqu'à ce que la condition soit vérifiée.		REPETER « traitement » JUSQU'A « condition » vraie
Boucle avec comptage On initialise la variable N avec une valeur x. On teste si N est égal à 0, si ce n'est pas le cas, on exécute le traitement et on décrémente la variable N puis on teste à nouveau la variable N, et ainsi de suite jusqu'à ce que N=0.		POUR N = x A 0 REPETER « traitement » FIN POUR

II. Graphe d'états

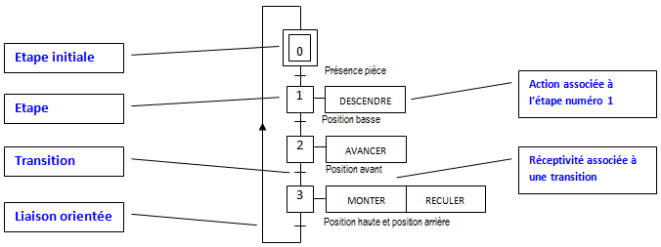
Les états peuvent être représentés graphiquement soit par des rectangles, soit par des ovales. Lorsqu'un état est composé de sous-états, on parle de super-état ou d'état composite.



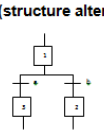
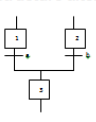
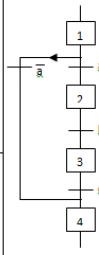
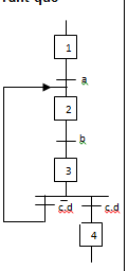
	**Etat OU :** ils représentent des états de fonctionnement mutuellement exclusif donc des états OU ne peuvent pas être actifs ou s'exécuter en même temps. On associe un nom à chaque état.
	**Etat ET :** ils représentent des états de fonctionnement totalement indépendants. Plusieurs états de même niveau hiérarchique peuvent être actifs simultanément. Ces états sont représentés graphiquement par un rectangle en trait pointillé, et un numéro indique l'ordre d'exécution. On associe un nom à chaque état.
	**Transitions :** Les transitions sont représentées par des flèches orientées, et permettent de décrire les évolutions du système d'un état source vers un état destination.
	**Transition par défaut :** Cette transition indique l'état (ou super-état) qui doit être actif à l'état initial (« mise sous tension »).
	**Actions dans un état :** Il s'agit de définir les actions à effectuer lorsque l'état *Nom-etat* est actif. On définit 3 types d'actions : - action à l'activation de l'état : pour spécifier ce type d'action, la syntaxe est *entry*: *actions*. - action durant l'état : pour spécifier ce type d'action, la syntaxe est *during*: *actions*. - action à la désactivation de l'état : pour spécifier ce type d'action, la syntaxe est *exit*: *actions*.

III. GRAFCET

Le GRAFCET est un outil graphique de description des comportements d'un système logique séquentiel. **Il est composé d'étapes, de transitions et de liaisons :**



Quelques structures utilisées :

Divergence en OU (structure alternative) :	Convergence en OU (structure alternative)	Boucle Si Alors	Boucle Répéter Tant que
			
Divergence en ET	**Convergence en ET**