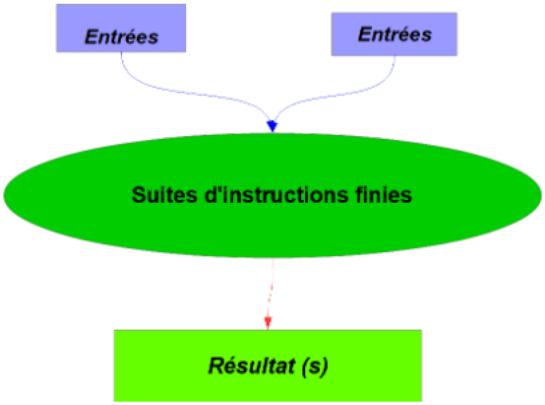
Algorithmique et programmation

Algorithme:

Définition:
Un algorithme est une démonstration structurée: on commence par justifer le problème puis on donne une suite d'instructions qui conduit à un résultat.
Algorithme = *suite finie et ordonnée d'instructions* qui indique la démarche à suivre pour résoudre un problème à partir des données de départ.
Il doit donc contenir des instructions *compréhensibles* et il est valide s'il est apte à réaliser exactement la tâche pour laquelle il a été conçu.

C'est une méthode qui utilise trois phases:	- Phase 1: Les entrées = données - Phase 2: Le traitement = suite finie d'instructions - Phase 3: Les sorties = résultat(s)
Explications:	- Phase 1= choix de la (ou des) valeur(s): informations à demander à l'utilisateur. - Phase 2= calculs à partir de cette (ou ces) valeur(s): liste d'instructions à appliquer aux données. - Phase 3 = restitution du résultat: le résultat peut être noté au fur et à mesure ou à la fin.

Recette:



Les entrées sont stockées dans des variables (qui correspondent à des tiroirs dans la mémoire pouvant contenir des objets (des valeurs)).
On donne un nom aux variables ce qui permet de les identifier et donc de les manipuler.

Exemple:

Processus de calcul:	Algorithme:	Explications:
	Variables *Réels: x, a, b, R* Début	On donne la liste des variables qui seront utilisées
1) Choisir un nombre 2) Lui ajouter 1 3) Elever au cube le résultat obtenu 4) Ajouter l'opposé de (-6) 5) Ecrire le résultat	*Saisir x* $a \leftarrow x+1$ $b \leftarrow a^3$ $R \leftarrow b+6$ Sortie *Afficher R*	On donne la liste des instructions: - récupérer des données - faire des affectations = étape qui consiste à mettre une valeur dans une variable. Afficher à l'écran les résultats obtenus

Le symbole $\leftarrow$ est celui de l'affectation

L'écriture d'un programme obéit d'une façon général à trois étapes importantes :
1) **Spécification: *LE QUOI***
2) **Conception: *LE COMMENT***
3) **Réalisation: *LA SOLUTION***

Dans l'étape de *spécification* ,on anlyse le problème en identifiant les données d'entrées et de sortie.
Dans l'étape *conception*, on s'intéresse à l'écriture des algorithmes et leur validation.
Dans la phase *réalisation*, on traduit les algorithmes en programmes, l'éxécution de ses derniers fournit la solution.

Exemple:
On veut calculer l'aire d'un champ rectangulaire:

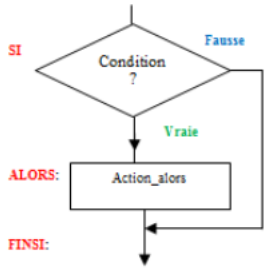
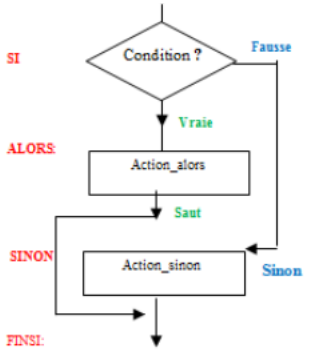
- **1ère étape: demander la longueur et la largeur du champ: on pose: a la longueur et b la largeur**
- **2nde étape: calculer l'aire: $s = a \times b$**
- **3ième étape: afficher le résulat : on affiche s**

*** Les deux réels "a" et "b" sont les données du problème = valeurs données par l'utilisateur**
*** La variable réelle "s" contient le résultat donné par l'algorithme**

Algorithme:
$a \longleftarrow lire$ $b \longleftarrow lire$ $s \longleftarrow a \times b$ écrire s

Instructions conditionnelles:

On va dire à l'algorithme d'exécuter des instructions que si une certaine condition est réalisée.
On va utiliser des tests: il y a deux structures possibles:

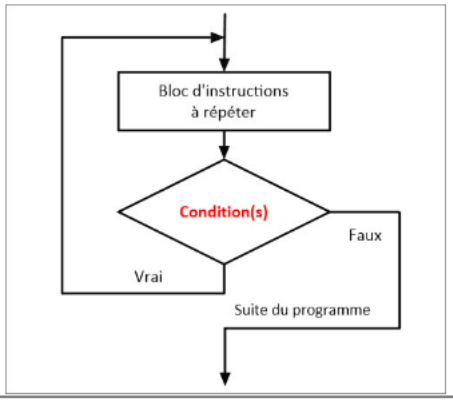
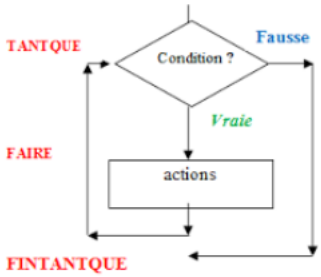
« si … alors … »	*« si … alors… sinon… »*
La condition est vraie, la suite d'instruction est exécutée. **Si la condition est fausse, on passe à la suite de l'algorithme.** 	**La condition ne doit offrir que deux réponses** **- La condition est vraie, on exécute l'instruction "action_alors"** **- La condition est fausse, on exécute l'instruction "action_sinon"** 

Exemple:
Trouver le plus grand de deux nombres donnés:

Variables ***Réels:** a, b* **Entrée** ***Saisir** a , b* **Traitement et sortie** **Si $a > b$ alors afficher "la plus grande valeur est", a** **Si $a < b$ alors afficher "la plus grande valeur est", b** **Si $a = b$ alors afficher "les deux valeurs sont égales"** **L'algorithme effectue les 3 comparaisons**	**Variables** ***Réels:** a, b* **Entrée** ***Saisir** a , b* **traitement et sortie** **Si $a > b$ alors afficher "la plus grande valeur est", a** **Sinon** **Si $a < b$ alors afficher "la plus grande valeur est",b** **Sinon** **Aficher "les deux valeurs sont égales"** **L'algorithme s'arrête si la 1ère comparaison est bonne**

Les boucles:

Une boucle permet de recommencer plusieurs fois un bloc d'instructions
Deux sortes de boucles sont possibles:

Boucles bornées:	*Boucles non bornées:*
On va demander à l'ordinateur de répéter une même tâche autant de fois que l'on souhaite. On exécute une boucle. La boucle va nous éviter de faire des calculs trop volumineux (exemple: faire la somme des 50 premiers chiffres) On va donc utiliser l'instruction « POUR…DE…A». On utilise cette boucle quand on connaît à l'avance le nombre de répétitions à répéter. On va donc nommer une nouvelle variable i, elle va nous indiquer le nombre de répétition. Cette dernière est un compteur, qui doit être déclarée au préalable comme les autres variables Une boucle « POUR …» est adaptée pour les calculs de somme ou pour les probabilités.  	Tant que « condition » faire « instruction ». L'idée est que tant que la condition est réalisée l'instruction est exécutée. On utilise cette boucle quand on veut répéter une série d'instructions sans que l'on sache à l'avance combien de fois. Mais il est indispensable de s'assurer que la condition du « TANT QUE… » finisse par être vérifiée (début du « tant que » et fin du « tant que… »). On teste si la condition est vraie: - Si elle est vraie: on exécute la suite d'instruction et on recommence. - Si elle est fausse: on s'arrête et on sort de la boucle. Une boucle « TANT QUE …» est adaptée pour les suites. 

Exemple:
Calcul du factoriel d'un entier N donné: $n! = 1 \times 2 \times \cdot \cdot \cdot \times n$.
$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$

Boucle "pour"	*Boucle "Tant que"*
Algorithme Factoriel Var N,I,F : entier Début Lire(N) F $\leftarrow$ 1 Pour I allant de 1 jusqu'à N Faire F $\leftarrow$ F $\times$ I FinPour Écrire(F) Fin	Algorithme Factoriel Var N,I,F : entier Début Lire(N) F $\leftarrow$ 1 I $\leftarrow$ 1 Tant que $(I \leqslant N)$ Faire F $\leftarrow$ F $\times$ I I $\leftarrow$ I $+$ 1 FinTantque Écrire(F) Fin

Conclusion:

Un algorithme est l'expression de la résolution d'un problème de sorte que le résultat soit calculable par une machine.
On a finalement l'enchaînement suivant :

Énoncé du problème $\longrightarrow$ *Algorithme* $\longrightarrow$ *Programme*