INITIATION AUX ALGORITHMES ET A LA PROGRAMMATION

FLAN

202 avenue de Colmar - 67100 STRASBOURG

303 88 30 78 30 30 30 88 28 30 69

selan@elan-formation.fr

www.elan-formation.fr

SAS ELAN au capital de 37 000 €
RCS Strasbourg B 390758241 – SIRET 39075824100041 – Code APE : 8559A

N° déclaration DRTEFP 42670182967 - Cet enregistrement ne vaut pas agrément de l'Etat

SOMMAIRE

I.	- 1	Notions de base de l'algorithmique	3
	1.	Définition	
	2.	Organigramme de programmation	3
	3.	Exemples de structures	4
	a.		
	b.	. Séguence alternative	5
	c.	Séquences répétitives	5
	d.	. Comment choisir la bonne boucle ?	6
	A	Les variables	-
	4.		
	a.		
	b.	71	
	c. d.		
	u.		
	5.	Les principaux opérateurs de traitement	8
	a.		
	b.	The same and the same production of the same and the same	
	c.	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
	6.	Insertion et affichage des données	9
II.	L	Les bases de la programmation : études d'algorithmes	10
	1.	Structure séquentielle	10
	2.	Les conditions – SI	11
	a.		
	b.		
	C.		
	3.	Les boucles – POUR, TANT QUE, JUSQU'A	
	a.		
	b.	Boucles complexes : TantQue & FaireJusqu'à	15
	C.	Boucles imbriquées	16
	1	Los tableaux	16

I. Notions de base de l'algorithmique

1. Définition

L'algorithmique est l'ensemble des règles et des techniques qui sont impliquées dans la définition et la conception d'algorithmes, c'est-à-dire de processus systématiques de résolution, par le calcul, d'un problème permettant de décrire les étapes vers le résultat.

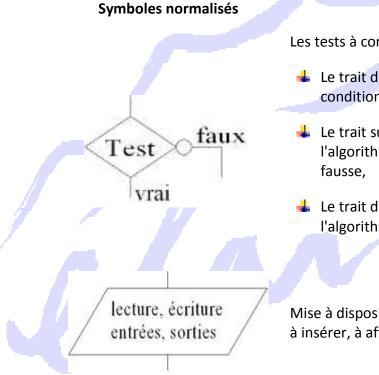
En d'autres termes, un algorithme est une suite finie et non-ambiguë d'opérations permettant de donner la réponse à un problème.

Si les opérations d'un algorithme s'exécutent les unes après les autres, on parle alors d'un algorithme séquentiel, si elles s'exécutent en parallèle, c'est un algorithme parallèle. Si l'algorithme exploite des tâches s'exécutant sur un réseau de processeurs, on parle alors d'algorithme réparti ou distribué.

2. Organigramme de programmation

On utilise souvent un organigramme (ou logigramme) pour représenter graphiquement un algorithme.

Une norme ISO (5807) décrit en détail les différents symboles à utiliser pour représenter un programme informatique de manière normalisée :

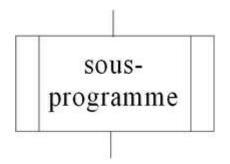


Commentaires

Les tests à conditions

- Le trait du haut représente l'entrée de la condition,
- Le trait sur le côté représente la suite de l'algorithme lorsque la condition est
- Le trait du bas représente la suite de l'algorithme si la condition est vérifiée.

Mise à disposition de différentes informations à insérer, à afficher, ...



Appel de sous-programme, de fonctions, ou de toutes autres structures externes.

On lit généralement et de manière conventionnel l'organigramme de haut en bas et/ou de gauche à droite. Si le cas nous l'impose, on peut éventuellement flécher le sens de lecture.

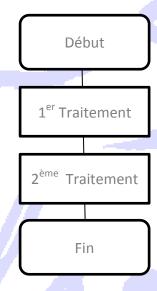
3. Exemples de structures

Il existe différentes structures suivant la complexité de l'algorithme et/ou du traitement à effectuer. On les appelle séquences de programmation.

a. Séquence linéaire

Séquence basique, elle permet une suite linéaire de l'algorithme, chaque traitement venant à la suite du précèdent, jusqu'à la fin de l'exécution.

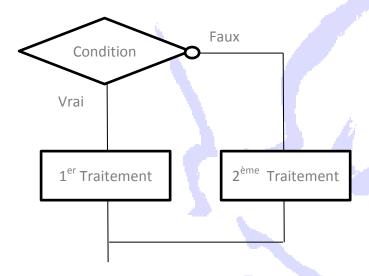
Organigramme



b. Séquence alternative

Structure conditionnelle, elle permet un traitement différent suivant le résultat d'une condition.

Organigramme



c. Séquences répétitives

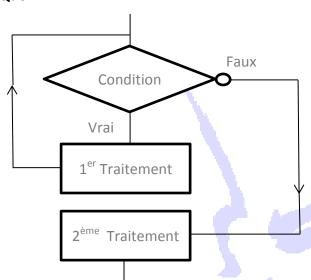
Aussi appelées boucles, elles permettent un traitement identique tant qu'une condition n'est pas vérifiée.

Il y a deux types de séquences :

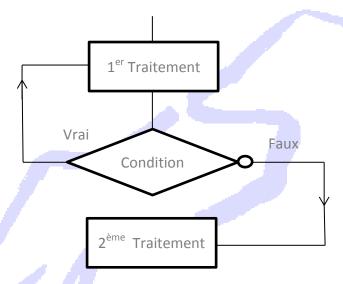
- **Tant que... faire...** : Tant que la condition est remplie, le traitement sera effectué.
- **Répéter...Jusqu'à...** : On refera le traitement tant que la condition de sortie de boucle ne sera pas remplie.

Organigrammes

♣ Tant…Que



♣ Répéter...Jusqu'à



d. Comment choisir la bonne boucle?

// A B	Nombre d'itération connu	Nombre d'itération inconnu		
La boucle doit être exécutée au moins une fois	TantQue	RépéterJusqu'à		
La boucle peut ne pas être exécutée	х	RépéterJusqu'à		

4. Les variables

Les variables peuvent être comparées à des morceaux de mémoires dans lesquelles les données peuvent être placées ou retirées à l'infini. En comparaison, on pourrait dire qu'une variable est comme une jarre, dans laquelle on peut soit verser de l'eau ou tout autre liquide, ou bien le retirer.

Elles sont créées et paramétrées au début de chaque algorithme, avec un nom et un type de valeur.

a. Nom

Le nom d'une variable doit obligatoirement commencer par une lettre et ne doit comporter aucun espace. On remplace souvent l'espace par un "tiret-bas" trouvé sous la touche [8] du clavier :

Tous les autres caractères sont interdits (signes logiques, opérateurs) ou déconseillés (accents).

Exemples

b. Types

Une variable peut contenir toutes sortes de données (texte, nombre), qu'elle soit quantitative comme un prix ou qualitative comme un code postal par exemple.

Suivant les données que l'on va mettre dans la variable, son type va changer entre numérique et textuel.

Il existe plusieurs types de données pour une variable, avec différents nom et limites d'utilisation, comme répertorié dans le tableau ci-dessous :

	Algorithme	Limites		
	Entier	Entier de -32 768 à 32 767		
Format nombre	Réel	Entier de -2 milliards à 2 milliards		
Format nombre	Simple	7 chiffres avec virgule flottante		
	Double	15 chiffres avec virgules flottante		
Booléen	Booléen	Vrai ou Faux		
Format texte	Chaine	Max. 64 000 caractères		
Date	Date	Aucune		
Tout type	Variable	Aucune		

c. Syntaxe

Lors de la déclaration d'une variable, son nom est précédé du mot **Var**, suivi du type de données.

Exemple

■ Var nom_variable: type de champ pour les algorithmes

d. Cas particulier: les constantes

Les constantes sont des variables spécifiques dans le sens où leurs contenus resteront inchangés tout au long du traitement, à l'inverse d'une variable qui verra son contenu être modifié.

Elles sont définis avec le mot **Const** précédent leur nom.

5. Les principaux opérateurs de traitement

Afin de permettre les calculs, les algorithmes utilisent différents opérateurs de traitement.

a. Opérateurs numériques

Opération	Opérateurs	Exemples
Addition	+	Longueur + Largeur
Soustraction	-	Prix – Remise
Division	/	Total / TotalGénéral
Multiplication	*	Prix * Quantité
Puissance /	۸	Rayon^2

b. Opérateurs de comparaison

Opération	Opérateurs	Exemples		
Supérieur	>	Age > 18		
Inférieur	'	Age < 18		
Inférieur ou égal	<=	Age <= 18		
Supérieur ou égal	>=	Age >= 18		
Différent	<>	Age <> 18		
Affectation d'un	= -	Total = Prix-Remise		
résultat				

c. Opérateurs logique

Opération	Opérateurs	Exemples
ET	And	Si (Test 1) And (Test 2) faire
OU	Or	Si (Test 1) Or (Test 2) faire
NON	Not	Si (Not Test 1) faire

6. Insertion et affichage des données

Il est possible (et conseillé) de prendre en compte l'interaction avec l'utilisateur lors de la conception d'un programme informatique.

Il y a deux types d'interaction possible : l'affichage et l'insertion des données. En algorithme on écrira Ecrire ou Lire.

Exemple

Algorithme
Lire (Longueur)
Ecrire (Surface)

Il est possible d'ajouter un texte dans les fonctions lire et écrire

↓ Lire ("Saisir votre nom : " ; Variable_Nom)

Il est possible de regrouper plusieurs information sur une même ligne à l'aide du symbole de concaténation &

• Ecrire ("Le chiffre d'affaires de la société :"; Nom_Client&" est de :"&CA_Client)

II. Les bases de la programmation : études d'algorithmes

1. Structure séquentielle

Structure linéaire basique, elle permet le traitement simple d'une programmation informatique.

Exemple 1: Calculer la surface d'un terrain

```
Algorithme Surface Terrain

Définition des variables

VAR var_longueur : Réel

VAR var_largeur : Réel

VAR var_surface : Réel

Début

// Saisie des données

Lire ("Saisir la longueur : " ; var_longueur)

Lire ("Saisir la largeur : " ; var_largeur)

// Traitement

var_surface = longueur * largeur

// Affichage du résultat

Ecrire ("La surface est de : "; var_surface)

Fin
```

Exemple 2 : Permuter deux valeurs

```
Algorithme Permutation

Définition des variables

VAR valeur1: Réel
VAR valeur2: Réel
VAR var_temp: Réel

Début

// Saisie des données
Lire ("Saisir la 1ère donnée: "; valeur1)
Lire ("Saisir la 2ème donnée: "; valeur2)

// Traitement

var_temp = valeur1
valeur1 = valeur2
valeur2 = var_temp

// Affichage du résultat
Ecrire ("Donnée 1 = : " & valeur1 & " Donnée 2 = " & valeur2)

Fin
```

2. Les conditions - SI

Les structures conditionnelles permettent un traitement différent suivant le résultat d'une condition (voir chapitre 3 b. Séquence alternative)

a. Condition simple

♣ SI...ALORS...

Elle se structure de la manière suivante : SI Condition ALORS Traitement FIN SI

Exemple: Calculer une remise sur chiffre d'affaire à 1 option

```
Algorithme Remise sur CA

Définition des variables

VAR CA: Simple

VAR Remise: Simple

Début

// Saisie des données

Lire ("Saisir le CA:"; CA)

// Traitement

SI CA > 10 000 Alors

Remise = CA * 0.10

FIN SI

// Affichage du résultat

Ecrire ("La remise est de:"; Remise)

Fin
```

SI...ALORS...SINON...

Elle se structure de la manière suivante : SI Condition ALORS Traitement 1 SINON Traitement 2 FIN SI

Exemple: Calculer une remise sur chiffre d'affaire à 2 options

```
Algorithme Remise sur CA 2
Définition des variables
   VAR CA: Simple
   VAR Remise: Simple
Début
// Saisie des données
   Lire ("Saisir le CA: "; CA)
// Traitement
     SI CA > 10 000 Alors
             Remise = CA * 0.10
     SINON
             Remise = CA * 0.05
     FIN SI
// Affichage du résultat
   Ecrire ("La remise est de : "; Remise)
Fin
```

b. Conditions imbriquées

Les conditions imbriquées permettent d'aller au-delà des 2 options de la condition simple.

Pour reprendre l'exemple de la remise sur le CA, nous allons rajouter une option à nos remises, ce qui donnera :

```
5% de remise si CA < 5 000€
10% de remise si CA > 5 000 € et CA < 10 000 €
15% de remise si CA > 10 000 €
```

Exemple: Calculer une remise sur chiffre d'affaire à 3 options

```
Algorithme Remise sur CA 3
Définition des variables
   VAR CA: Simple
   VAR Remise: Simple
Début
// Saisie des données
   Lire ("Saisir le CA: "; CA)
// Traitement
SI CA < 5 000 Alors
             Remise = CA * 0.05
   SINON
     SI CA > 10 000 Alors
             Remise = CA * 0.10
     SINON
             Remise = CA * 0.05
     FIN SI
FIN SI
// Affichage du résultat
    Ecrire ("La remise est de : "; Remise)
Fin
```

c. Conditions multiples: ET, OU

Les conditions multiples permettent d'être plus précis lors de la condition, de rajouter des conditions, etc. ...

Elles se structurent de la manière suivante :

```
SI (condition 1 ET condition 2) ALORS...SINON...FIN SI
SI (condition 1 OU condition 2) ALORS...SINON...FIN SI
```

Attention, le **ET** est restrictif: il faut remplir favorablement les deux conditions pour pouvoir vérifier le test, alors que le **OU** n'en nécessite qu'une seule des deux, ou les deux.

Exemple de problématique : Nous devons attribuer 3 jours de congés pour toutes les femmes qui ont 2 enfants ou plus, 2 jours de congés pour tous les autres.

Exemple : Calculer les jours de congés

```
Algorithme Calcul congés
Définition des variables
   VAR sexe : Texte
   VAR Nb Enfants: Réel
   VAR conges : Réel
Début
// Saisie des données
   Lire ("Saisir le sexe : "; sexe)
   Lire ("Saisir le nombre d'enfants : "; Nb_Enfants)
// Traitement
   SI sexe = "F" ET Nb_Enfants >= 2 ALORS
             conges = 3
   SINON
             conges = 2
   FIN SI
// Affichage du résultat
   Ecrire ("Le nombre de congés attribué est de : "; conges)
Fin
```

3. Les boucles – POUR, TANT QUE, JUSQU'A

a. Boucles simples: Pour i = 1 à n

Il existe des travaux répétitifs dont les traitements sont toujours les mêmes mais appliqués à des données différentes.

Dans ce cas il est possible de programmer des boucles de traitement.

La boucle **For compteur = 1 à n** permet de répéter un traitement autant de fois que le compteur **n** n'est pas atteint.

Exemple: Calculer la puissance d'un nombre

```
Algorithme Calcul puissance
Définition des variables
    VAR Nombre : Réel
    VAR Puissance: Réel
    VAR i : Réel
    VAR Resultat : Réel
Début
// Saisie des données
    Lire ("Saisir le nombre : "; Nombre)
    Lire ("Saisir la puissance : "; Puissance)
// Traitement
    Resultat = Nombre * Nombre
    Pour i = 2 à Puissance
             Resultat = Resultat * Nombre
    Fin Pour
// Affichage du résultat
    Ecrire ("Le résultat est : "; Resultat)
Fin
```

b. Boucles complexes: Tant...Que & Faire...Jusqu'à

Tant...Que

Le test de boucle est positionné avant l'exécution du traitement, qui sera réalisé tant que la condition de la boucle est vraie.

Exemple : Inscrire 10 livres (Attention, nécessite de lire **le chapitre 4 : les tableaux** au préalable)

```
Algorithme Saisie livre par 10

Définition des variables

VAR Nom_Livre : chaine (1 à 10)

Var Auteur_Livre : chaine (1 à 10)

VAR i : entier

Début

// Saisie des données

i = 1

Tant que i <=10 Faire

Lire ("Saisir le nom du livre : " ; Nom_Livre(i))

Lire ("Saisir le nom de l'auteur : " ; Auteur_Livre(i))

i = i +1

Fin Tant Que

Fin
```

👃 Faire...Jusqu'à

Le test de boucle est placé après le traitement. Le travail est réalisé jusqu'à ce que la condition de boucle soit remplie, il est donc fait **au moins une fois** à l'inverse des boucles précédentes.

c. Boucles imbriquées

Comme pour les structures conditionnelles, plusieurs boucles peuvent être imbriquées les unes dans les autres.

Exemple: Calculer le total des salaires

```
Algorithme Calcul salaires
Définition des variables
VAR Salairemensuel: Réel 'Salaire mensuel par salarié
VAR Salairecumule: Réel 'Salaire cumulé par salarié
VAR Salaire total : Réel 'Salaire total de tous les salariés
VAR i : réel 'Indice 'Nombre de salarié
VAR m : réel 'indice des mois
Début
// Saisie des données & Traitement
   Salairecumule = 0
   Salairetotal = 0
   Pour i = 1 à 5
      Pour m = 1 à 6
         Lire ("saisir le salaire mensuel : "; Salairemensuel)
         Salaire cumule = Salairecumule+Salaire mensuel
      Fin pour
      Ecrire ( " Le salaire total du salarié est égal à : "; Salairecumule)
      Salairetotal = Salairetotal + Salairecumule
      Salaire cumulé = 0
   Fin pour
// Affichage du résultat
    Ecrire ("Le total des salaires du semestre est : "; Salairetotal)
Fin
```

Remarque : A chaque itération, la valeur contenue dans le champ Nombre est multiplié par lui-même.

4. Les tableaux

Lorsque l'on gère une grande quantité d'informations, il devient laborieux de devoir saisir chacune de ces données dans une variable différente. Il existe donc une méthode beaucoup plus pratique : les tableaux.

Le principe du tableau est de créer pour chaque donnée un champ qui s'incrémentera à chaque nouvelle information, comme pour une base de données, et permettra d'appeler la valeur référencée.

Exemple: Tableau des livres

Donnée	Science-Fiction	Roman	BD	Polar	Biographie
Indice tableau	1	2	3	4	5

→ Le type de livre 3 (indice du tableau) est BD

Comment déclarer un tableau?

Le champ est de type chaine et le nombre d'éléments de la chaine est précisé entre parenthèse à la suite du type de champ :

VAR Mon_tableau : chaine(1 à 5) → Le tableau contiendra 5 éléments de type chaine de caractère.

Exemple : Saisie de données dans un tableau

```
Algorithme Saisie tableau

Définition des variables

VAR Mon_Tableau : Chaine ( 1 à 5 )

Début

// Saisie des données

Lire ("Saisir la 1ère donnée : " ; Mon_Tableau(1))

Lire ("Saisir la 2ème donnée : " ; Mon_Tableau(2))

Lire ("Saisir la 3ème donnée : " ; Mon_Tableau(3))

Lire ("Saisir la 4ème donnée : " ; Mon_Tableau(4))

Lire ("Saisir la 5ème donnée : " ; Mon_Tableau(5))

Fin
```