Secteur Tertiaire Informatique
Filière « Etude et développement »

Séquence « Ecrire un Algorithme »

Algorithmique

Apprentissage

Mise en pratique

Evaluation

TABLE DES MATIERES

Objectifs

Etre capable d'écrire un **Algorithme** et de le mettre en œuvre à partir du **pseudocode** dans **AlgoBox**.

Pré requis

Aucun.

Outils de développement

« AlgoBox »

Méthodologie

Etude initiale de ce support de cours.

Approfondissement grâce aux ressources supplémentaires.

1. UN ALGORITHME, QU'EST-CE QUE C'EST ?

Définition : Un **algorithme** est une **suite finie d'instructions**, qui une fois **exécutée** correctement, conduit à un résultat donné.
Pour fonctionner, un algorithme doit donc contenir uniquement des instructions compréhensibles par celui qui devra l'exécuter.

Un algorithme est en fait une « recette de cuisine ».

La maîtrise de l'algorithmique requiert 2 qualités :

- Il faut une certaine **intuition**, car aucune recette ne permet de savoir à priori quelles instructions permettront d'obtenir le résultat voulu.
- Il faut être **méthodique** et **rigoureux**. En effet, chaque fois qu'on écrit une série d'instructions qu'on croit justes, il faut systématiquement se mettre à la place de la machine qui va les exécuter, pour vérifier si le résultat obtenu est bien celui que l'on voulait.

Les **ordinateurs**, quels qu'ils soient, sont fondamentalement capables de comprendre que **4** catégories d'**instructions** seulement. Ces 4 catégories d'instructions sont :

- L'**affectation de variables**
- La **lecture** / **écriture**
- Les **Tests**
- Les **Boucles**

Apprendre à écrire un algorithme, c'est apprendre à manier la **structure logique** d'un **programme informatique**.

2. CONVENTIONS POUR ECRIRE UN ALGORITHME.

Historiquement, plusieurs types de **notations** ont représenté des **algorithmes**.

Il y a eu notamment une **représentation graphique**, avec des carrés, des losanges, etc. … qu'on appelait des **organigrammes**. Mais dès que l'algorithme commence à grossir, ce n'est plus pratique du tout.

On utilise généralement une série de **conventions** appelée « **pseudo-code** », qui ressemble à un **langage de programmation** authentique dont on aurait évacué la plupart des problèmes de syntaxe. Mais ce **pseudo-code** est purement conventionnel et n'est pas interprété par aucune machine.

3. LES VARIABLES

Définition : Dès que l'on a besoin de stocker une information au cours d'un programme, on utilise une **variable**.

Pour employer une image, une **variable** est une **boîte**, que le programme va repérer par une **étiquette**. Pour avoir accès au contenu de la boîte, il suffit de la désigner par son **étiquette**.

Avant de pouvoir **utiliser une variable**, il faut **créer la boîte** et lui **coller une étiquette**, c'est ce qu'on appelle la **déclaration de variables**.

Le **nom de la variable** (l'étiquette de la boîte) obéit à des impératifs changeant selon les langages de programmation.

Un nom de variable peut comporter des **chiffres** et des **lettres**, mais il exclut la plupart des signes de ponctuation et en particulier les espaces. Un **nom de variable** correct **commence** également par une **lettre**.

Une fois le nom de la variable choisi, il faut déterminer le **type de la variable**. On distingue **3 types** usuels de variable :

- Type **alphanumérique** : du texte.
- Type **numérique** : un nombre (entier, décimal, réel).
- Type **liste ou tableau** : ensemble de nombres ordonné.

La seule chose qu'on puisse faire avec une variable, c'est **l'affecter**, c'est-à-dire lui **attribuer une valeur**.

a ← 8 Attribue la valeur **8** à la variable **a**

a ← b Attribue la valeur de **b** à la variable **a**

c ← a + 6 Attribue la valeur **a+6** à la variable **c**

b ← b +1 Incrémente de **1** la valeur de la variable **b**

Les **opérateurs** sont de **2** sortes **Opérateurs numériques** et **Opérateurs logiques** :

Opérateurs numériques		**Opérateurs logiques**	
Addition	+	**ET**	Intersection de 2 ensembles
Soustraction	-	**OU**	(non exclusif) union de 2 ensembles
Multiplication	*	**Non**	Complémentaire d'un ensemble
Division	/		
Puissance	^		

4. LECTURE ET ECRITURE D'UNE VARIABLE

Définition : **Lire** ou **Entrer** une variable signifie que l'utilisateur doit rentrer une valeur pour que le programme la lise.

Ecrire ou **Afficher** une variable signifie que le programme renvoie la valeur de la variable qu'il a trouvé.

5. LES TESTS

Il y a deux formes pour un test : soit la **forme complète**, soit la **forme simplifiée** :

Forme complète	Forme simplifiée
Si condition **alors** Instructions 1 **Sinon** Instructions 2 **FinSi**	**Si** condition **alors** Instructions **FinSi** (Si la condition n'est pas vérifiée, le programme ne fait rien).

La condition portant sur une variable peut être :

- Une valeur à atteindre
- Une comparaison avec une autre variable (égalité, inégalité, non égalité)

On peut aussi mettre un test qui se décompose en plusieurs conditions reliées par un opérateur logique.

- Condition 1 **ET** condition 2 : Les deux conditions doivent être vérifiées en même temps.
- Condition 1 **OU** condition 2 : Au moins l'une des deux conditions doit être vérifiée.

On peut aussi imbriquer un ou plusieurs autres tests à l'intérieur d'un test. On a alors le cheminement suivant :

Si condition 1 **alors**
    Instructions 1
Sinon
        Si condition 2 **alors**
            Instructions 2
        Sinon
            Instructions 3
        FinSi
FinSi

6. LES BOUCLES

Définitions : Une **boucle** est une structure **répétitive** ou **itérative**, c'est-à-dire que la boucle effectue **n fois** un calcul sous le contrôle d'une **condition d'arrêt**.

La **boucle conditionnelle** obéit au schéma suivant :

Tant que condition
    Instructions
FinTantque

La **boucle en comptant**
Si on connaît à l'avance le nombre d'incrémentations, on a la structure suivante :

Pour compteur = initial **à** final **par valeur du pas**
    Instructions
Fin Pour

7. LES FONCTIONS

Une application, surtout si elle est longue, a toutes les chances de devoir procéder aux mêmes traitements, ou à des traitements similaires, à plusieurs endroits de son déroulement.

Afin **d'éviter** la **répétition de code,** ce qui doit être une préoccupation constante du développeur, il a possibilité d'exécuter un morceau de code à l'intérieur d'un bloc. Ce bloc sera soit une **procédure** ou soit une **fonction**.

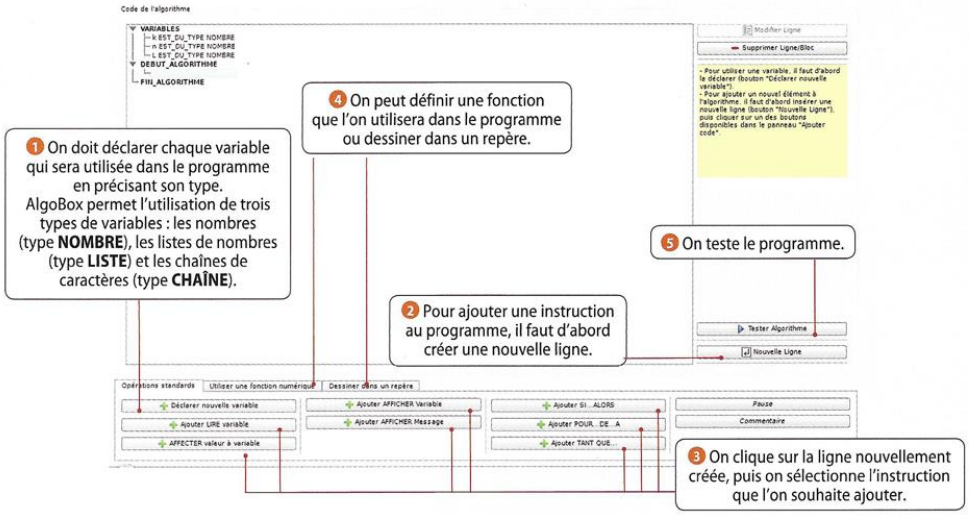
Dans notre étude des algorithmes avec **AlgoBox**, nous parlerons exclusivement de **fonctions**, et nous apprendrons à les écrire et à les appeler au sein d'un même programme.

Exemple d'utilisation d'une fonction pour le calcul récursif de la **factorielle** d'un nombre entier positif n.

```
1  FONCTIONS_UTILISEES
2    FONCTION factorielle(param)
3      VARIABLES_FONCTION
4      DEBUT_FONCTION
5      SI (param == 0) ALORS
6        DEBUT_SI
7        RENVOYER 1
8        FIN_SI
9      RENVOYER param*factorielle(param-1)
10     FIN_FONCTION
11 VARIABLES
12   n EST_DU_TYPE NOMBRE
13   resultat EST_DU_TYPE NOMBRE
14   monmessage EST_DU_TYPE CHAINE
15 DEBUT_ALGORITHME
16   LIRE n
17   resultat PREND_LA_VALEUR factorielle(n)
18   AFFICHER "La Factorielle est :"
19   AFFICHER resultat
20 FIN_ALGORITHME
```

8. PROGRAMMER AVEC ALGOBOX

Programmation avec AlgoBox



Calculs et fonctions

Racine carrée de x	sqrt(x)
x à la puissance n	pow(x,n)
π	Math.PI
Partie entière de x	floor(x)
Nombre aléatoire compris entre 0 et 1	random()
Entier aléatoire compris entre a et b	ALGOBOX_ALEA_ENT(a,b)
Valeur absolue de x	abs(x)
P(X = k) si X suit la loi B(n, p)	ALGOBOX_LOI_BINOMIALE(n,p,k)

Expressions conditionnelles

Tester si $x = 1$	x==1
Tester si $x \neq 1$	x!=1
Tester si $x \leqslant 1$	x<=1
Tester si $x \geqslant 1$	x>=1
Tester si $1 < x < 3$	x>1ETx<3
Tester si $x = 3$ ou $x = 5$	x==3OUx==5

Copier, coller et couper une ligne ou un bloc de l'algorithme

Avec le menu Edition, il est possible de copier/coller/couper :

– une ligne du type
...PREND_LA_VALEUR..., AFFICHER...

– un bloc du type
POUR ... DE ... A, SI ... ALORS et TANT QUE...

Pour copier et couper tout le contenu de ce type de bloc, il faut se placer sur la première ligne du bloc en question.

⚠ Pour **coller** une ligne ou un bloc de code, il faut d'abord créer une nouvelle ligne.

Commandes de boucles

Instructions	Une fois la boîte de dialogue ouverte	Remarques
SI ... ALORS	On saisit une expression conditionnelle.	On peut ajouter l'instruction **SINON** en cochant la case correspondante dans la fenêtre de dialogue.
POUR ... DE ... A	On choisit la variable de type **NOMBRE** utilisée grâce au menu déroulant et on renseigne les valeurs de début et de fin.	La valeur de cette variable est automatiquement augmentée de 1 à chaque boucle.
TANT QUE ...	On saisit une expression conditionnelle.	La condition peut porter sur une ou plusieurs variables des trois types **NOMBRE**, **CHAÎNE** ou **LISTE**.

9. EXEMPLES D'ALGORITHMES

- **1 - L'Algorithme d'Euclide** (Calcul du **PGCD** de deux nombres entiers naturels)

  Effectuer la division euclidienne de **a** par **b** et noter **r** le reste ;

  Remplacer **a** par **b** ;

  Remplacer **b** par **r** ;

  Recommencer les calculs précédents jusqu'à ce qu'une division donne un reste égal à 0.

  Le PGCD de a et b est le dernier reste non nul.

  Ecrire ce petit programme en utilisant **AlgoBox**

- **2 – L'Algorithme de la Factorielle** (non récursif)

  Variables : **k**, **i**, **n** entiers

  Entrées et initialisation

  Lire **k**

  n ← 1

  Traitement

  Pour **i** de **1** à **k** faire

  n ← n*i

  FinPour

  Sortie : Afficher **n**

  Ecrire ce petit programme en utilisant **AlgoBox**

- **3 – Le jeu d'échec et les grains de riz**

  Une légende raconte que l'inventeur du jeu d'échec présenta le jeu à son roi qui, enthousiaste, lui demanda ce qu'il désirait en récompense.

  L'inventeur demanda simplement un grain de riz sur la première case, deux sur la deuxième, quatre sur la troisième, huit sur la quatrième, et ainsi de suite en multipliant à chaque fois le nombre de grains par deux.

  Sachant qu'un échiquier comporte 64 cases, écrire l'algorithme qui donne le nombre de grains de riz total sur l'échiquier.

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
Alexandre RESTOUEIX - Formateur

Date de mise à jour : 10/06/2019