Un peu d'algorithmique :

1 Introduction

1.1 Les instructions

Un algorithme, c'est une succession d'instructions. Voici par exemple un algorithme très simple pour résoudre l'équation $6x + 7 = 0$:

- Soustraire 7 à chaque membre de l'équation
- Diviser chaque membre de l'équation par 6
- Afficher la solution de l'équation

A faire : Exécuter cet algorithme.

1.2 La structure conditionnelle

On l'utilise lorsque l'on fait une étude de cas. C'est une structure du type suivant :

Structure d'algorithme 1. Structure conditionnelle

Si *Condition*
  Alors $Instruction_1$
    $Instruction_2$
    ...
  Sinon $Instruction_a$
    $Instruction_b$
    ...
Fin_Bloc_Si

Exemple. Algorithme 1.1 Préparer son cartable

Si *Cours de mathematiques*
  Alors *Prendre la calculatrice*
  Sinon *Ne pas la prendre*
Fin_Bloc_Si

Exemple. Algorithme 1.2 Signe du produit $f(x) \times g(x)$

Si $Signe(f(x)) == Signe(g(x))$
  Alors $Signe(produit) = " + "$
  Sinon $Signe(produit) = " - "$
Fin_Bloc_Si

A faire. Algorithme 1.3 Equation de la droite (AB)

Si $x_A = x_B$
  Alors

  Sinon

Fin_Bloc_Si
$Afficher\ la\ droite$

1.3 La structure itérative

On l'utilise lorsque l'on doit répéter plusieurs fois une instruction. C'est une structure qu'on peut rencontrer sous plusieurs formes.

1.3.1 Forme "Pour Chaque"

Structure d'algorithme 2.a Structure itérative "Pour Chaque"

Pour Chaque *Element dans un ensemble donne*
Faire *Instruction*$_1$
Instruction$_2$
...
Fin_Bloc_Pour_Chaque

Exemple. Algorithme 2.a Préparer son cartable

Pour chaque *Matiere du lendemain*
Faire *Prendre le cahier de la matiere*
Prendre le livre de la matiere
Fin_Bloc_Pour_Chaque

1.3.2 Forme "Tant Que"

Structure d'algorithme 2.b Structure itérative "Tant Que"

Tant Que *Condition*
Faire *Instruction*$_1$
Instruction$_2$
...
Fin_Bloc_Tant_Que

Exemple. Algorithme 2.b Comportement en classe

Tant Que *Le professeur ou un camarade a la parole*
Faire *Ecouter la personne*
Prendre en note sur son cahier
Fin_Bloc_Tant_Que

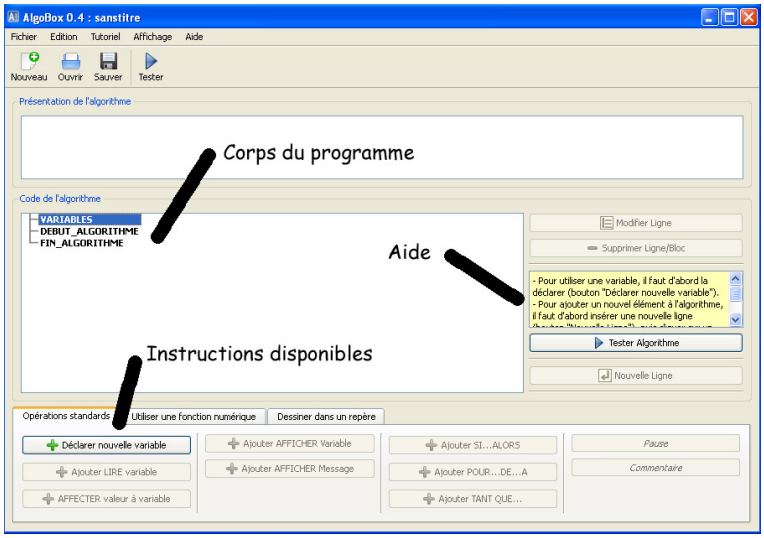
1.4 L'algorithme de Syracuse

On rappelle l'algorithme de Syracuse :

Algorithme 3 Suite de Syracuse

Choisir un nombre N
Tant Que $N > 1$
Faire **Si** *N est pair*
Alors $N = \frac{N}{2}$
Sinon $N = 3 * N + 1$
Fin_Si
Afficher N
Fin_Bloc_Tant_Que

2 Algobox



Le programme de départ comporte 3 lignes (voir ci-contre)

a) Déclaration des variables

Il faut expliquer à l'ordinateur quelles variables vont servir. Pour cela, toute variable que l'on va utiliser doit être déclarée en cliquant sur le bouton 

. Une fois déclarée, elle apparait en-dessous de la première ligne "Variables"

b) Écriture du programme

Le programme proprement dit doit être écrit entre "Début_Algorithme" et "Fin_Algorithme". Pour ajouter une instruction au programme, il est nécessaire de créer d'abord une nouvelle ligne en cliquant sur le bouton 

. La nouvelle ligne est insérée en-dessous de la ligne actuellement sélectionnée.

c) Exécution du programme

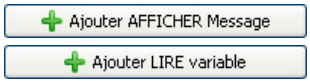
Cliquer sur 

: une fenêtre s'ouvre. Cliquer sur 

.

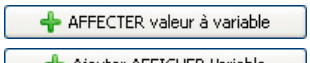
2.1 Notre premier programme : la moitié d'un nombre

et taper "N".



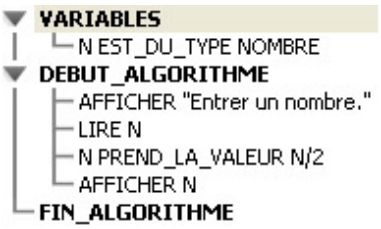
et taper "Entrer un nombre". [a]

et taper "N". [b]



et taper "N/2".

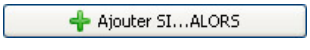
et taper "N".



a. Ceci sert à afficher un message lors de l'exécution de l'algorithme. On peut cocher la case "Ajouter un retour à la ligne" pour que les affichages soient à chaque fois écrits à la ligne les uns des autres. Il est conseillé de le faire !

b. Ceci sert à demander à l'utilisateur de taper un nombre lors de l'exécution du programme et de l'affecter à la variable N

2.2 Tester si un nombre est pair

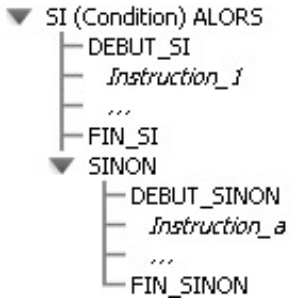
Effectuer un test, c'est utiliser une structure conditionnelle (Si... Alors...). On ajoute une structure conditionnelle avec 

Lorsque l'on veut ajouter une instruction à faire dans le cas "Sinon", il ne faut pas oublier de cocher "Ajouter SINON" lors de la création de la structure "Si... Alors..."

Pour tester si un nombre est pair, on rentre "FLOOR(N/2) == N/2" dans la condition de la structure "Si... Alors...". Floor désigne la fonction partie entière. Effectivement, un nombre est pair si et seulement si lorsqu'on le divise par 2, on obtient un nombre entier.

Correspondance Algorithme français / Algorithme algobox.

Si	*Condition*	
	Alors	$Instruction_1$
		$Instruction_2$
		...
	Sinon	$Instruction_a$
		$Instruction_b$
		...
Fin_Bloc_Si		



A faire : Ecrire un programme qui lit un nombre au clavier, qui teste s'il est pair, et qui affiche "Ce nombre est un entier pair" si le nombre est pair ou "Ce nombre n'est pas un entier pair" sinon.

2.3 L'algorithme de Syracuse

Correspondance Algorithme français / Algorithme algobox.

Tant Que	*Condition*	
	Faire	$Instruction_1$
		$Instruction_2$
		...
Fin_Bloc_Tant_Que		

A faire : Taper sous Algobox l'algorithme de Syracuse. Le tester sur plusieurs exemples.

Pour ceux qui ont fini : Compter le nombre d'opérations que le programme effectue avant d'obtenir la valeur 1.

Pour cela, on va utiliser une variable "Compteur". Au début du programme, l'algorithme n'a encore effectué aucun calcul. Il faut donc initialiser "Compteur" à 0 (soit, en langage Algobox, "Compteur reçoit la valeur 0")

Ensuite, à chaque fois que l'on rentre dans la boucle "Tant Que", on rajoute 1 à la variable compteur (soit, en langage Algobox, "Compteur reçoit la valeur Compteur + 1")

Tout à la fin de l'algorithme, après la boucle Tant Que, on peut alors afficher la valeur de la variable "Compteur".