Introduction à Algobox

Document complet sur :

http://www.xm1math.net/algobox/tutoalgobox/index.html

I. Introduction

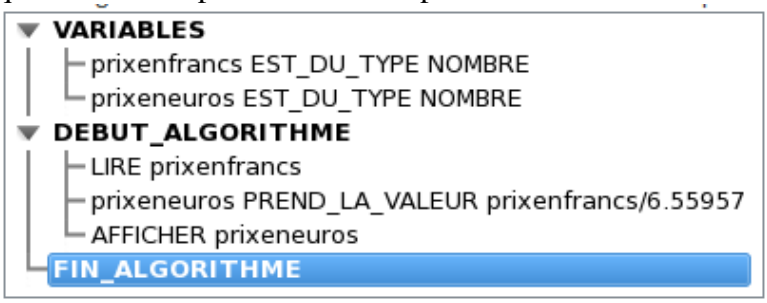
Premières règles concernant la conception d'un algorithme avec AlgoBox :

- Il faut toujours commencer par déterminer les variables nécessaires à la bonne marche de l'algorithme et indiquer leurs noms à **AlgoBox** en utilisant le bouton `Déclarer nouvelle variable`. Une variable ne pourra être utilisée par **AlgoBox** que si elle est préalablement déclarée.
- Pour que l'utilisateur puisse entrer des données, il faut utiliser le bouton `Ajouter LIRE variable` qui permettra à l'utilisateur de donner une valeur à la variable sélectionnée.
- Pour pouvoir donner une valeur à une variable (après un calcul éventuellement) à l'intérieur de l'algorithme, il faut utiliser le bouton `AFFECTER valeur à variable`. La boîte dialogue qui apparaît permet de sélectionner la valeur à laquelle on veut affecter une valeur et l'expression (ou le calcul) qui permet d'obtenir cette valeur.
- Pour pouvoir afficher un résultat correspondant à la valeur d'une variable, il faut utiliser le bouton `Ajouter AFFICHER variable` et sélectionner la variable en question dans la boîte de dialogue qui apparaît.
- Pour ajouter un nouvel élément à l'algorithme (autre que la déclaration d'une variable), il faut d'abord insérer une nouvelle ligne en se positionnant à l'endroit adéquat et en cliquant sur le bouton `Nouvelle Ligne`.

Remarque : pour repartir d'un algorithme vide dans **AlgoBox**, il suffit de cliquer sur le bouton `Nouveau` en haut du programme (il est alors demandé à l'utilisateur s'il veut d'abord enregistrer l'algorithme courant)

Exemple :

On écrit un algorithme qui converti un prix en francs en prix en euros :



Activité :

Ecrire un algorithme qui donne l'aire d'un rectangle dont on connait la longueur et la largeur.

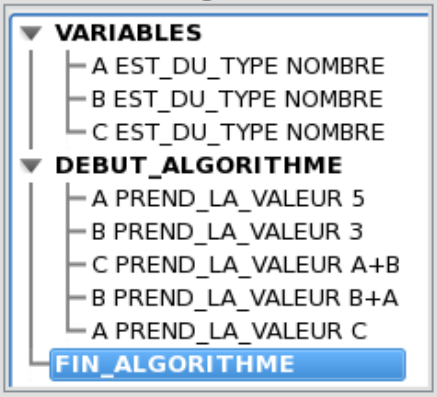
II. Variables et affectations

Remarques générales à propos des variables qui, comme nous l'avons déjà vu, servent à stocker des données :

- Le nom des variables ne doit pas inclure d'espaces, ni de caractères spéciaux.
- La valeur d'une variable peut changer au fil des instructions de l'algorithme.
- Les variables informatiques peuvent servir à stocker des données de différents types, mais pour le moment nous nous contenterons d'utiliser des variables du type `NOMBRE` (type par défaut dans **AlgoBox**)

Activité :

On considère l'algorithme ci-dessous :



Déterminer quelles sont les valeurs des variables A, B et C à la fin de l'exécution de l'algorithme.

Conclusion :

- Les opérations sur les variables s'effectuent ligne après ligne et les unes après les autres.
- Quand l'ordinateur exécute une ligne du type "*mavariable* PREND_LA_VALEUR *un calcul*", il effectue d'abord *le calcul* et stocke **ensuite** le résultat dans *mavariable*.

Activité :

On considère l'algorithme ci-dessous :

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    y EST_DU_TYPE NOMBRE
    z EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE x
    y PREND_LA_VALEUR x-2
    z PREND_LA_VALEUR -3*y-4
    AFFICHER z
FIN_ALGORITHME
```

Si on entre x = 5, qu'affiche cet algorithme ?

III. Instructions conditionnelles

Le test « **Si … alors … sinon** »

```
SI (une certaine condition est vérifiée) ALORS
    DEBUT_SI

    FIN_SI
```

ou bien

```
SI (une certaine condition est vérifiée) ALORS
    DEBUT_SI

    FIN_SI
SINON
    DEBUT_SINON

    FIN_SINON
```

Exemple :

Calcul de la racine carrée :

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    racine EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE x
    SI (x>=0) ALORS
        DEBUT_SI
        racine PREND_LA_VALEUR sqrt(x)
        AFFICHER racine
        FIN_SI
FIN_ALGORITHME
```

Détermine le plus grand de deux nombres :

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    y EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE x
    LIRE y
    SI (x>y) ALORS
        DEBUT_SI
        AFFICHER x
        FIN_SI
    SINON
        DEBUT_SINON
        AFFICHER y
        FIN_SINON
FIN_ALGORITHME
```

Exercice :

Concevoir un algorithme dans lequel on demande à l'utilisateur d'entrer un nombre et s'il est différent de 1, l'algorithme doit afficher la valeur de y = 1/(x-1).

Rmq : x différent de 1 se code « x!=1 »).

IV. Les boucles

Pour… de … a :

Exemple :

Donner la racine carrée de tous les entiers compris entre 1 et 50 :

```
VARIABLES
    n EST_DU_TYPE NOMBRE
    racine EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    POUR n ALLANT_DE 1 A 50
        DEBUT_POUR
        racine PREND_LA_VALEUR sqrt(n)
        AFFICHER racine
        FIN_POUR
FIN_ALGORITHME
```

Remarques importantes sur les boucles `POUR...DE...A` dans **AlgoBox** :

- La variable servant de compteur pour la boucle doit être du type NOMBRE et doit être déclarée préalablement (comme toutes les variables).
- Dans **AlgoBox**, cette variable est automatiquement augmentée de 1 à chaque fois.
- On peut utiliser la valeur du compteur pour faire des calculs à l'intérieur de la boucle, mais **les instructions comprises entre** `DEBUT_POUR` et `FIN_POUR` **ne doivent en aucun cas modifier la valeur de la variable qui sert de compteur**.

Activité :

Ecrire un algorithme calculant : 1 + 1/2 + 1/3 + … + 1/10

Tant que …

La structure :

```
TANT_QUE (une certaine condition est vérifiée) FAIRE
    DEBUT_TANT_QUE

    FIN_TANT_QUE
```

Remarques importantes sur les structures `TANT QUE...` dans **AlgoBox** :

- On utilise cette structure quand on veut répéter une série d'instructions sans que l'on sache à l'avance combien de fois *(quand on connaît exactement le nombre de répétitions à effectuer, on utilise plutôt une boucle `POUR...DE...A`)*
- Si la condition du `TANT QUE...` est fausse dès le début, les instructions entre `DEBUT_TANT_QUE` et `FIN_TANT_QUE` ne sont jamais exécutées *(la structure TANT QUE ne sert alors strictement à rien)*.
- Il est indispensable de s'assurer que la condition du `TANT QUE...` finisse par être vérifiée (le code entre `DEBUT_TANT_QUE` et `FIN_TANT_QUE` doit rendre vraie la condition tôt ou tard), sans quoi l'algorithme ne pourra pas fonctionner. Cette structure est donc à manier avec prudence...

Exemples :

On rembourse 2500 € sans intérêt, quel est le montant du dernier versement ?

```
VARIABLES
    montant EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    montant PREND_LA_VALEUR 2500
    TANT_QUE (montant>=110) FAIRE
        DEBUT_TANT_QUE
        montant PREND_LA_VALEUR montant-110
        FIN_TANT_QUE
    AFFICHER montant
FIN_ALGORITHME
```

On cherche le plus petit entier n tel que : $2^N > 10000$

(`pow(2,N)` est le code AlgoBox pour calculer 2^n) :

```
VARIABLES
    N EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    N PREND_LA_VALEUR 1
    TANT_QUE (pow(2,N)<10000) FAIRE
        DEBUT_TANT_QUE
        N PREND_LA_VALEUR N+1
        FIN_TANT_QUE
    AFFICHER N
FIN_ALGORITHME
```

Activité :

On lance une balle d'une hauteur initiale de 300 cm. On suppose qu'à chaque rebond, la balle perd 10 % de sa hauteur. On cherche à savoir le nombre de rebonds nécessaire pour que la hauteur de la balle soit inférieure ou égale à 10 cm.