UNE APPROCHE DE L'ALGORITHMIQUE

L'objectif est de rendre les élèves capables :
· de décrire certains algorithmes en langage naturel,
· d'en réaliser quelques uns (tableur ou petit programme sur calculatrice),
· d'interpréter quelques algorithmes plus complexes.

Le logiciel utilisé pour ces exemples est le logiciel **ALGOBOX** c'est un logiciel très simple et très rapide de prise en main qui me semble parfaitement adapté à la découverte de l'algorithmique.
Vous trouverez l'aide complète de ce logiciel est sur le site http://algo.jeanlepine.com/
Vous trouverez tous les renseignements sur le site du logiciel ALGOBOX

Je propose ici quelques acitivités progressives qui permettent à un élève de seconde de découvrir les premières bribes de l'algorithmique comme prévu dans le programme de maths. L'activité autour du jeu du "c'est plus c'est moins" me paraît une très bonne activité pour découvrir les principes de l'algorithmique car elle est très ouverte et permet aux élèves de faire de nombreux tests et de découvrir de nombreuses solutions. Il ne faut pas hésiter à utiliser le mode "pas à pas" du logiciel Algobox car il permet de voir évoluer les variables et permet de faire un parfait débogage.
Chacune des activités proposées commence par la mise en place d'un algorithme suivi de quelques exercices.

Pour prolonger ces activités et faire de la programmation à un autre niveau avec des constructions de procédures de fonctions etc, il faudra utiliser des langages standards comme le visual basic ou le pascal avec la plateforme Delphi ou bien encore le C avec C++.

Sommaire

I* découverte de l'algorithmique : la division

Commençons cette première approche par un problème de calcul tout simple.
Il s'agit d'entrer deux nombres entiers A et B et de récupérer le quotient Q de ces deux nombres.

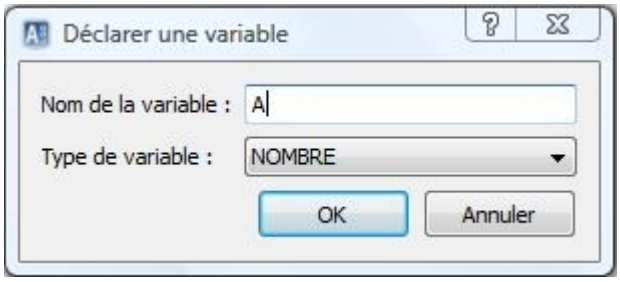
Méthode :
Demander la saisie du nombre A
Demander la saisie du nombre B
Calculer dans Q le quotient A/B
Afficher la valeur de Q

Variables utilisées
trois variables numériques A, B, Q

Réalisation de l'algorithme

1* Lancer le logiciel **Algobox**

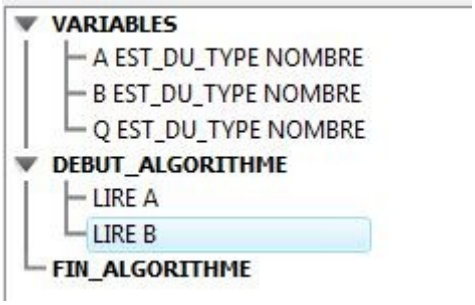
2* Cliquer sur déclarer une nouvelle variable



3* Taper la variable A
en laissant bien *nombre*
comme *type de variable*.

4* Cliquez alors sur le bouton **Ajouter une ligne** ou appuyez sur la combinaison de touches [*Ctrl] [Entrée*] puis cliquez sur le bouton **lire variable** pour demander la *saisie du nombre A*

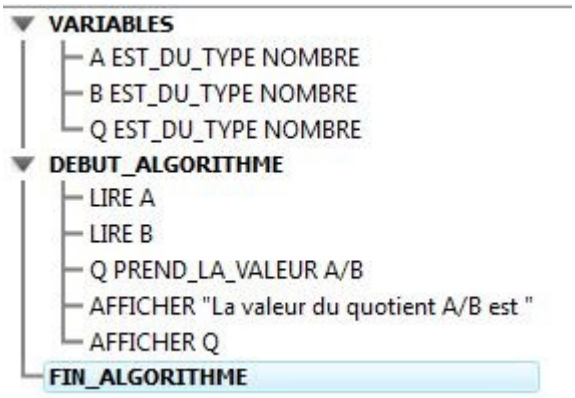
5* Faire la même opération pour la *saisie du nombre B*.
A cette étape votre algorithme doit être comme ci-contre.
Il nous faut, maintenant que les deux nombres A et B sont en mémoire, calculer le quotient en priant un peu pour que le nombre B ne soit pas nul !



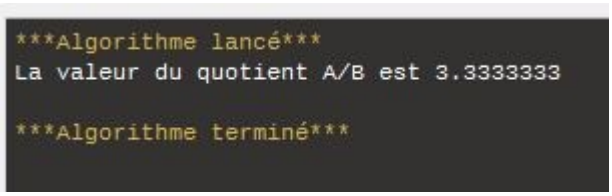
6* Cliquez sur **Nouvelle ligne** puis sur le bouton **Affecter valeur à variable**.
Choisir la variable **Q** et à la suite de *prend la valeur* taper **A/B**.
Il nous reste à afficher le quotient obtenu

7* Générer une ligne puis cliquez sur **ajouter afficher message** et taper le message ***La valeur du quotient A/B est.*** *Ajouter une ligne* puis cliquez alors sur **ajouter afficher variable** et choisir **Q** en ajoutant un r*etour à la ligne.*

L'algorithme doit alors être celui-ci.



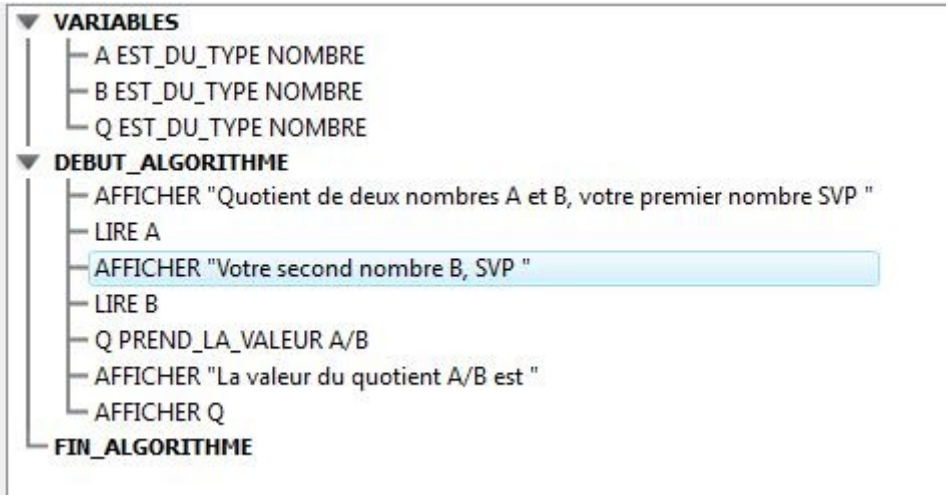
Vous pouvez alors faire une sauvegarde de votre travail en faisant **Fichier Sauver sous.**



8* Il s'agit maintenant de tester notre algorithme. Cliquer sur le bouton **tester** puis sur le bouton **lancer l'algorithme.** Le résultat s'affiche comme dans la fenêtre ci-contre après avoir tapé les nombres 10 et 3.

Quelques améliorations dans l'algorithme.

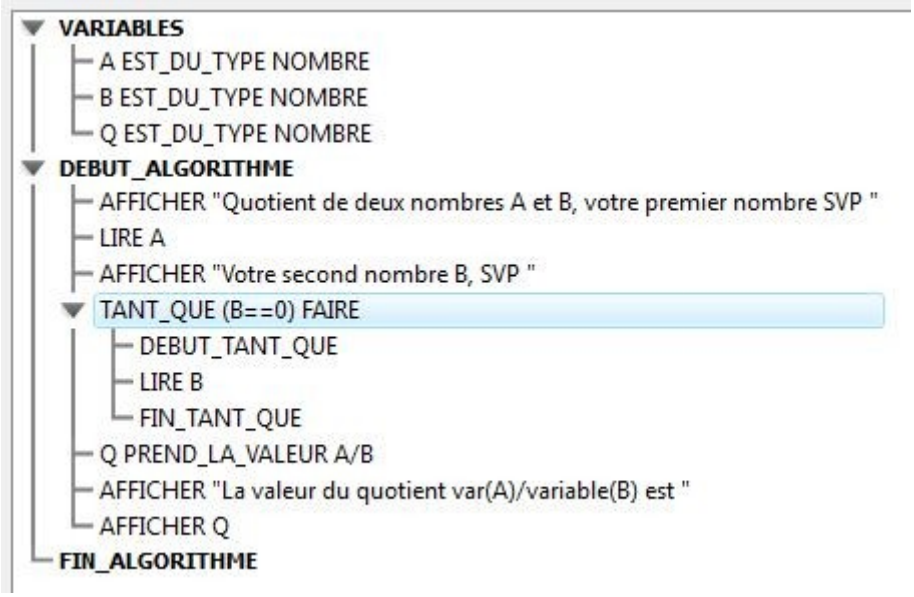
Il serait agréable d'avoir une petite explication avant de saisir les deux nombres afin d'éviter de taper un diviseur égal à zéro !!!
Vous pouvez utiliser le bouton **Nouvelle ligne** pour insérer deux lignes comme ci-contre et le bouton **Afficher message** pour obtenir l'algorithme ci-contre.



La structure algorithmique *Tant que*

Il est assez risqué de laisser l'utilisateur saisir un diviseur égal à zéro !
Pour cela nous allons demander de lire B *tant que la valeur de B est de zéro*.
Méthode : i**nsérer une ligne** après la ligne **AFFICHER « Votre second nombre B, SVP »** puis cliquer sur le bouton **Ajouter Tant que.** Comme condition taper **B==0**, en faisant bien attention de mettre **deux signes =** ! Ce double = pour indiquer que nous ne sommes pas sur une affectation de variable mais sur un test d'égalité : *la variable B est-elle égale à zéro* ?

Entre les deux balises DEBUT et FIN ajouter la ligne LIRE B et supprimer l'ancienne ligne.
Vous obtenez alors l'algorithme ci-contre.



Il ne reste qu'à faire le test en essayant de taper la valeur zéro pour le diviseur du quotient.

Nouvelle amélioration : fixer le nombre de décimales à afficher.
Ce problème est mathématique.
De la réponse *3,257257257* il faut arriver à *3,25* par exemple !
En fait, une solution simple consiste à utiliser la fonction **partie entière** en procédant de la façon suivante *3,257257257*100* devient *325,7257257* dont on prend la **partie entière** *325*. Il suffit alors de **diviser le résultat par 100** pour obtenir *3,25* !
La fonction **partie entière**, comme dans de nombreux langages informatiques, est la fonction **floor()**.

Notre ligne de calcul devient donc celle-ci

Q = floor(A/B*100)/100

Au niveau de l'algorithme il suffit de se positionner sur la ligne ***Q prend la valeur A/B*** puis de cliquer sur le bouton **Modifier ligne** !
Remplacer alors **A/B** par la formule **floor(A/B*100)/100**

Tester l'algorithme pour vérifier.

*Vos premiers essais : à vous de chercher, modifier et tester. Quand l'algorithme ne fonctionne pas comme vous le désirez n'hésitez pas à utiliser le mode **pas à pas** d'**algobox** pour voir comment évolue le contenu de toutes vos variables.*

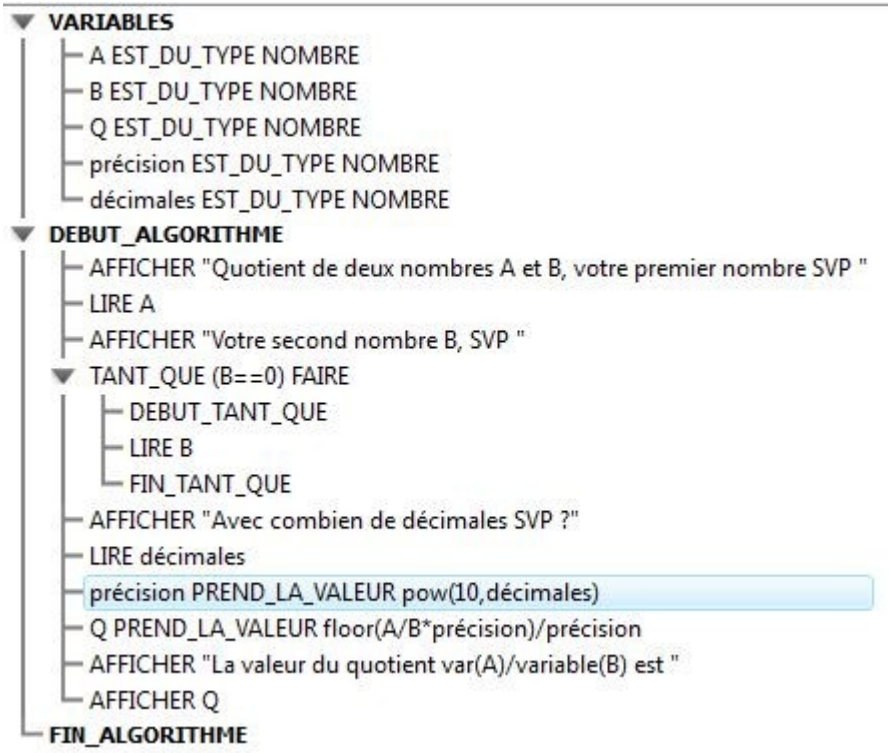
Exercice 1 : Modifier l'algorithme pour obtenir un affichage à 3 décimales.

Exercice 2 : Il s'agit de rajouter une variable pour demander à l'utilisateur le nombre de décimales souhaité.
Note pour cet exercice : il faut utiliser la fonction puissance **pow(x,n)** qui correspond à la puissance nieme de x,

Exercice 1 : Modifier l'algorithme pour obtenir un affichage à 3 décimales.

Exercice 2 : Il s'agit de rajouter une variable pour demander à l'utilisateur le nombre de décimales souhaité.
Note pour cet exercice : il faut utiliser la fonction puissance **pow(x,n)** qui correspond à la puissance nieme de x,

Solutions



Rigolo : la division à l'ancienne.
Les premiers processeurs ne faisaient que des additions et soustractions ! Comment alors réaliser une division ?

Voici la méthode directe employée pour faire la division entière de deux nombres A et B.

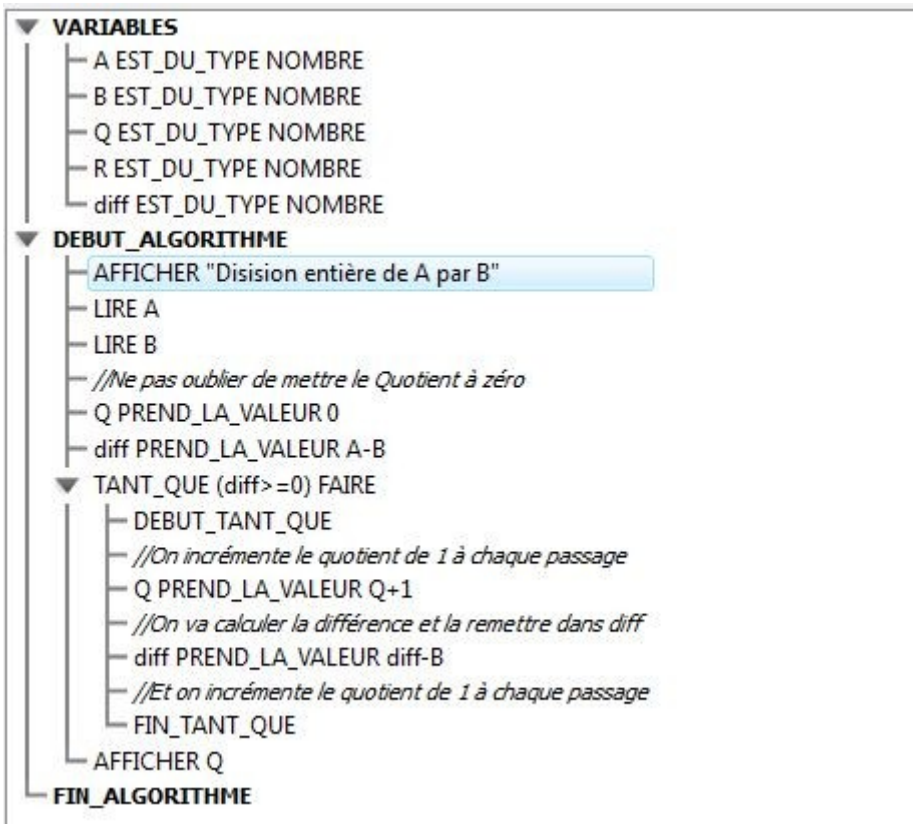
On soustrait B de A tant que c'est possible et on compte le nombre de soustractions faites.

C'est aussi simple que cela !

Exemple : soit à chercher le quotient entier de 11 par 3
on calcule 11 – 3 = 8 et on compte 1
puis 8-3 = 5 et on compte 2
puis 5- 3 = 2 et on compte **3**
puis 2-3 impossible dans IN donc on s'arrête de compter !

Le quotient de 11 par 3 est donc **3** !

Ci-contre l'algorithme.



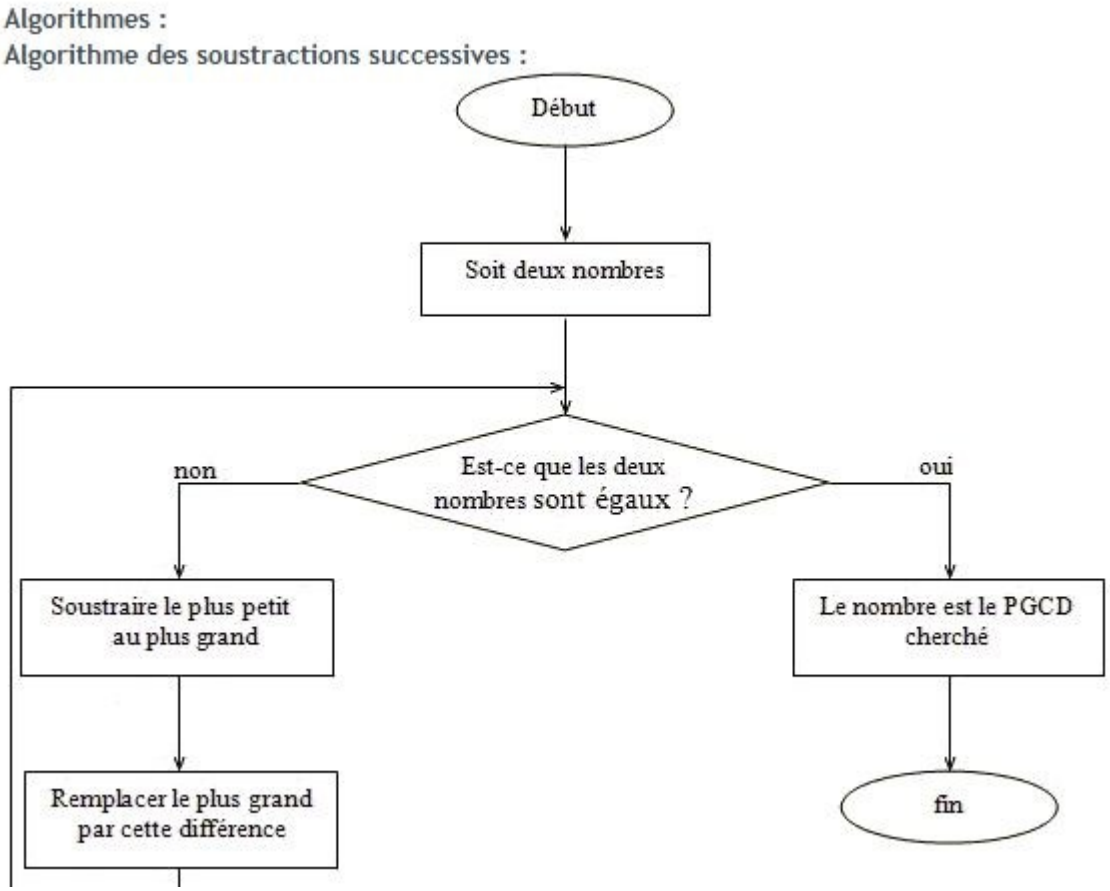
Exercice 3 : que se passe-t-il si les lignes **Q PREND LA VALEUR Q+1** et d**iff PREND LA VALEUR diff-B** sont inversées ? L'algorithme fonctionne-t-il encore correctement ?

Exercice 4 : Utiliser la variable R pour afficher le reste dans la division de A par B à la place du quotient Q

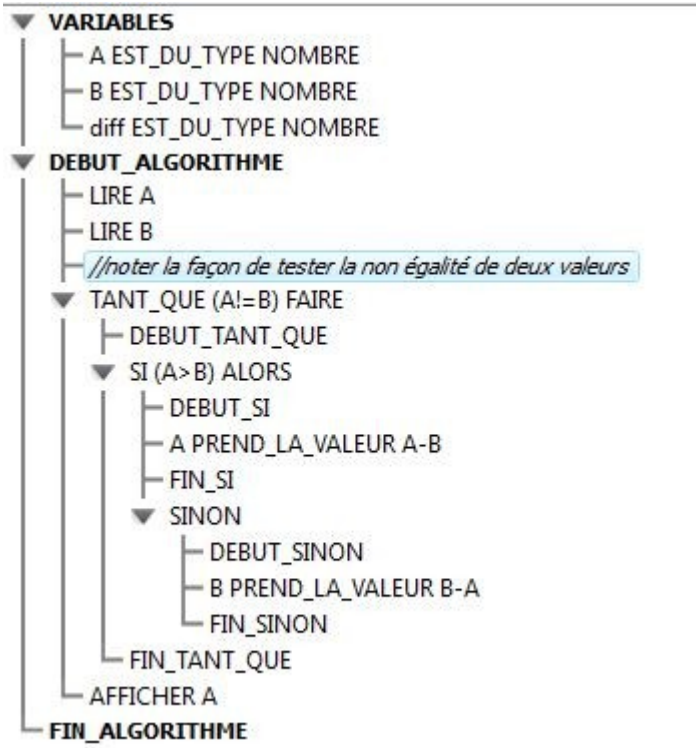
II – Le pgcd de deux nombres entiers

Rappelons que le plus grand diviseur commun aux deux nombres **12** et **18** est **6** car
les diviseurs de **12** sont {1; 2; 3; 4; 6; 12}
les diviseurs de **18** sont {1; 2; 3; 6; 9; 18},
Les diviseurs communs aux deux sont {1; 2; 3; 6 }
et le plus grand est donc **6** !

Voici ci-dessous un algorithme pour déterminer le PGCD de deux nombres entiers A et B



Algorithme correspondant avec **algobox**



Exercice : utiliser cet algorithme pour déterminer si deux nombres sont premiers entre eux.
On rappelle que pour que deux nombres soient premiers entre eux il suffit que leur PGCD soit 1.

III- Travailler sur quatre semaines

Dans certains emplois du temps il faut découper sur quatre semaines plutôt que sur deux. C'est le cas pour certains élèves de seconde en informatique dans notre lycée,
En semaine 41 nous sommes en semaine 1
En semaine 42 nous sommes en semaine 2
En semaine 43 nous sommes en semaine 3
En semaine 44 nous sommes en semaine 4
En semaine 45 nous sommes en semaine 1
etc.

On voudrait faire afficher, à partir du numéro de semaine du calendrier, le numéro de semaine de l'emploi du temps (1, 2, 3 ou 4)

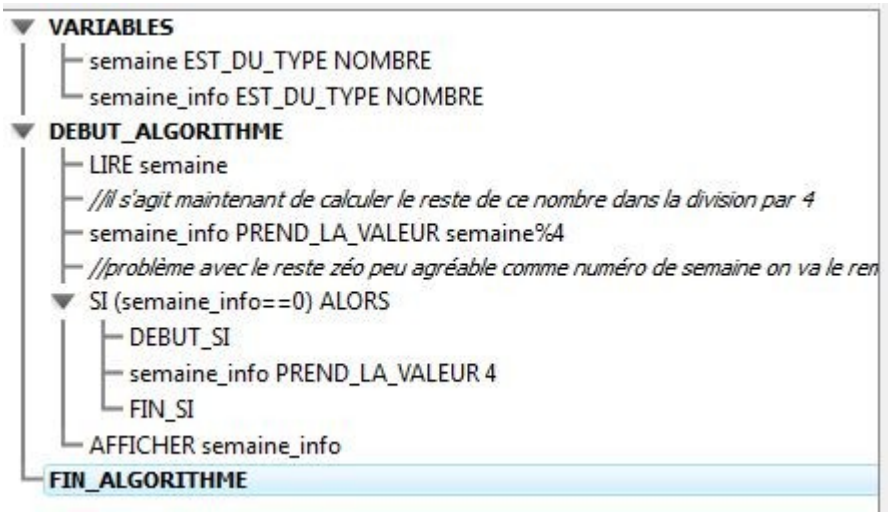
Méthode :

Prendre le numéro de semaine du calendrier dans la variable ***semaine***
Calculer le reste dans la division entière par 4 de cette **semaine**
et le stocker dans ***semaine_info***
Si ***semaine_info*** vaut zéro le mettre à 4

Les variables à utiliser seront donc
semaine, de type numérique
semaine_info, de type numérique aussi.

*Il faudra utiliser la fonction **A%B** qui donne le reste dans la division euclidienne de A par B.*

Algorithme



Exercice 1
On sait que les cours d'informatique de la classe de 2C sont en semaine 4.
Écrire un algorithme permettant d'afficher toutes les semaines 4 de l'année.

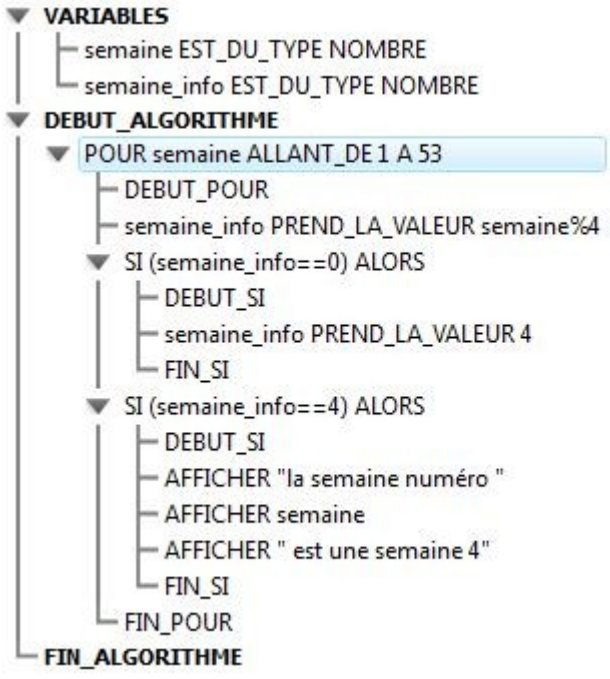
Il faudra donc utiliser une boucle pour explorer les 53 semaines de 1 à 53
La boucle aura la structure suivante.

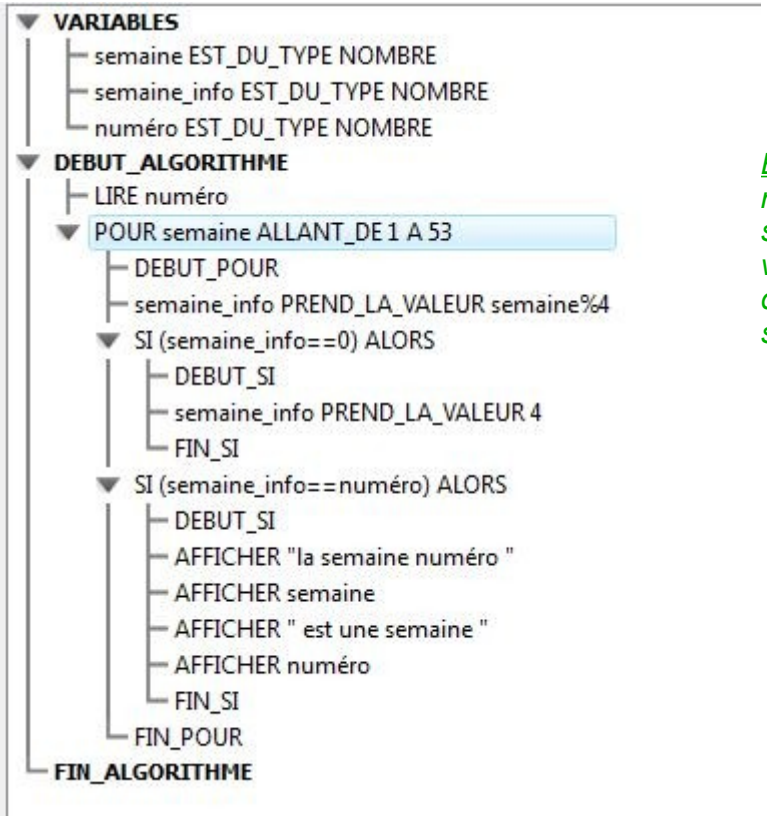
POUR semaine ALLANT DE 1 A 53
DEBUT POUR
*
*
*
FIN POUR

Exercice 2 : la même problématique, mais cette fois pour une des quatre semaines au choix de l'utilisateur. On va créer une nouvelle variable et demander à l'utilisateur quelle liste de semaines il veut , 1, 2, 3 ou 4.

Solutions

Exercice 1
On sait que les cours d'informatique de la classe de 2C sont en semaine 4.
Écrire un algorithme permettant d'afficher toutes les semaines 4 de l'année.





Exercice 2 : la même problématique, mais cette fois pour une des quatre semaines au choix de l'utilisateur. On va créer une nouvelle variable et demander à l'utilisateur quelle liste de semaines il veut , 1, 2, 3 ou 4.

IV- Avec des chaines de caractères : les palindromes

*Le travail avec les variables alphanumériques dites **chaînes de caractères** est toujours assez difficile en programmation. Les mots, les phrases, les structures n'obéissant pas du tout à des règles mathématiques mais étant bourrée d'exceptions.*

Ici, il s'agit de créer un algorithme **qui renverse un mot** ou une expression pour vérifier s'il s'agit d'un palindrome (comme LAVAL, ERDRE ou RADAR).

Méthode :
Un mot est saisi dans la variable ***CHAINE* palind**
La variable ***CHAINE* result** va récupérer une par une les lettres de **palind** à partir de la dernière etc.

Il est utile pour cet algorithme de connaître les *fonctions de manipulation de chaines* suivante :

palind.*length* renvoi la longueur de la chaine (**5** si vous avez tapé **laval**)

palind.*substr(3,1)* renvoie la chaine de longueur 1 située en 3ieme position (**v** pour **laval**)

*Attention : en déclarant les variables il faut choisir le type **chaine**.*

Voici l'algorithme :

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    palind EST_DU_TYPE CHAINE
    result EST_DU_TYPE CHAINE
    longueur EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    //Vidons la chaine result
    result PREND_LA_VALEUR ""
    LIRE palind
    //On a besoin de la longueur de la chaine palind
    longueur PREND_LA_VALEUR palind.length
    POUR x ALLANT_DE 1 A longueur+1
        DEBUT_POUR
        //Le dernier caractère est mis en premier dans result puis l'avant dernier en second etc
        result PREND_LA_VALEUR result+palind.substr(longueur-x+1,1)
        FIN_POUR
    AFFICHER result
FIN_ALGORITHME
```

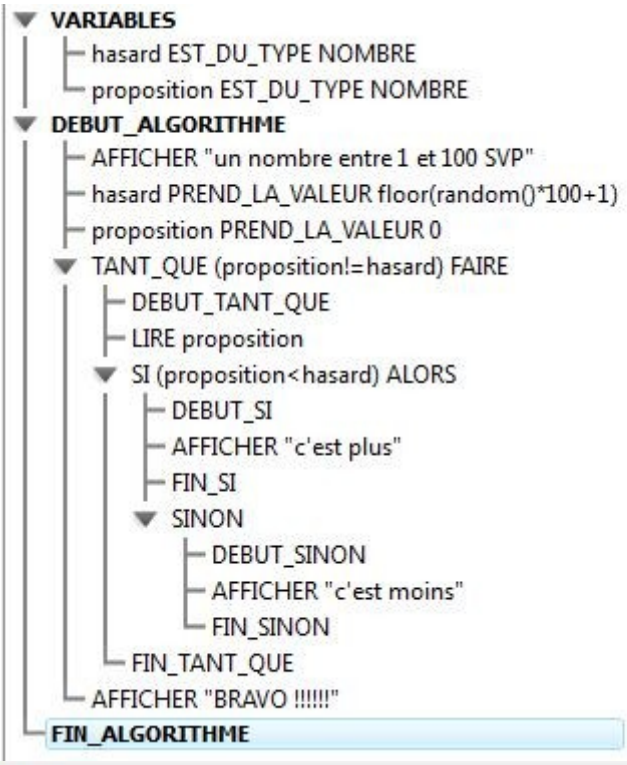
Exercice : compléter l'algorithme ci-dessus en ajoutant un test **SI ALORS** pour afficher un *message de réussite* si le mot tapé est bien un *palindrome*.

V- Le jeux du c'est plus c'est moins

Classique jeu du c'est plus c'est moins avec un nombre entre 1 et 100 choisi au hasard par la machine.

Pour calculer un nombre au hasard entre 1 et 100 on utilise la fonction **random()** qui renvoie une valeur décimale entre ***0 et 0,9999999*** !
La formule classique pour obtenir un nombre entre 1 et 100 sera la suivante **floor(random()*100+1)**

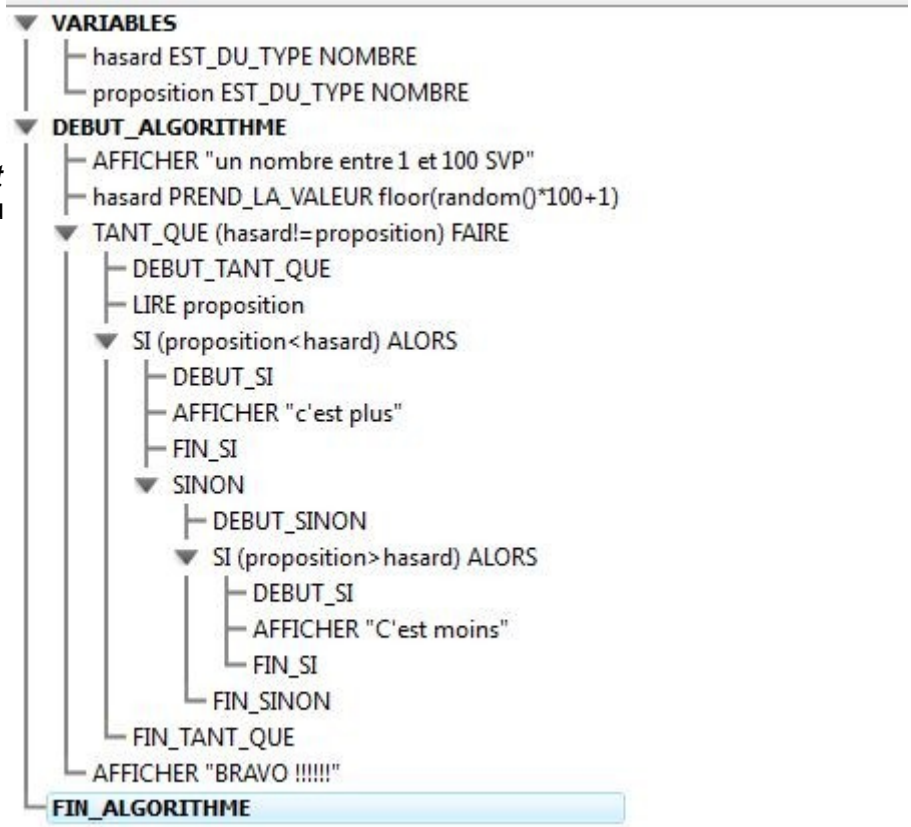
Explications :
random() sort un nombre décimal entre 0 et 0,99999999
random()*100 donne donc un décimal entre 0 et 99,99999
random()*100+1 donne un décimale entre 1 et 100,99999
il ne reste plus qu'à prendre la partie entière floor() !!!

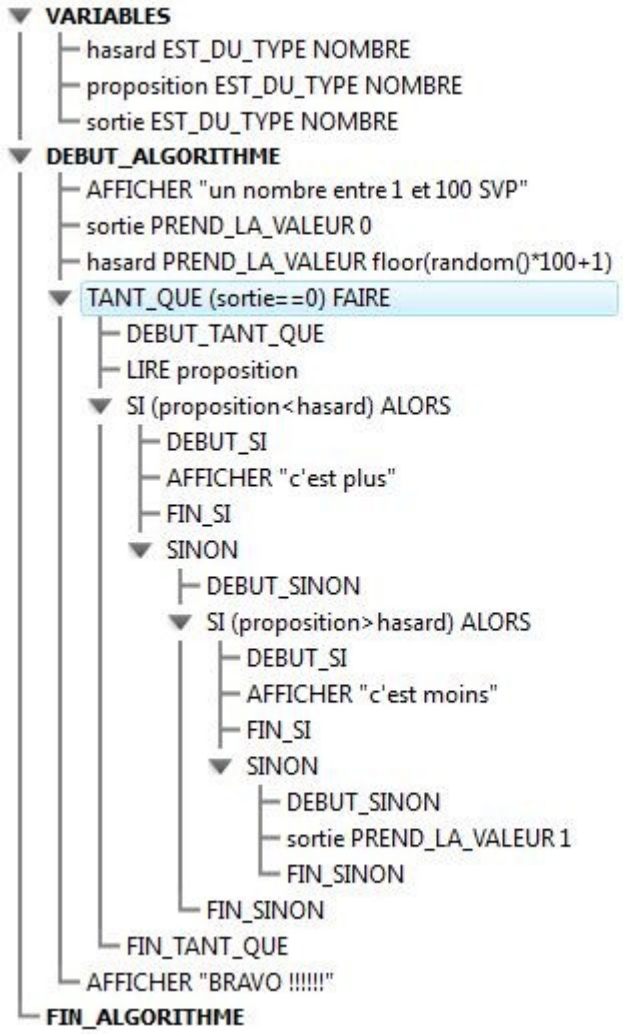


Algorithme ci-contre

Nous avons là un problème algorithmique très intéressant car nous utilisons une instruction **TANT QUE** dans laquelle nous avons **un test** (proposition!=hasard) *en début de boucle* alors qu'il aurait fallu ce test *en fin de boucle* pour ne pas avoir à faire un passage inutile !!! Si bien que nous nous trouvons souvent avec un ***« c'est moins »*** en trop.

Nous devons donc ajouter **un test dans le SINON** pour empêcher l'affichage du « ***c'est moins*** » en cas d'égalité... l'algorithme devient donc celui-ci





passage d'une boucle.

Problème qui aurait pu être solutionné aussi avec une variable **sortie** qui prend la valeur 1 en cas d'égalité comme dans l'algorithme ci-contre.

Lorsque vous modifiez votre algorithme vous pouvez utiliser les racouçis claviers ci-après pour les lignes ou les blocs
[Ctrl] [C] pour copier
[Ctrl] [X] pour couper
[Ctrl] [V] pour coller

*N'oubliez pas dans les recherches ci-dessous d'utiliser la fonction **pas à pas** qui vous permet de surveiller la valeur de vos variables à chaque*

Exercice 1 :*En dernière ligne proposer à l'utilisateur le nombre de coups qui ont été nécessaires avec un affichage du genre :*
BRAVO, vous avez trouvé en 7 coups !

Exercice 2 :
Afficher l'intervalle dans lequel se trouve le nombre pour faciliter la tache à l'utilisateur !
Par exemple : après avoir proposé le nombre 50 l'utilisateur verra s'afficher un message du genre :
C'est moins, le nombre est situé entre 0 et 50
puis après avoir proposé 20
C'est plus, le nombre est situé entre 20 et 50
etc

Solutions

Exercice 1 :*En dernière ligne proposer à l'utilisateur le nombre de coups qui ont été nécessaires avec un affichage du genre : BRAVO, vous avez trouvé en 7 coups !*

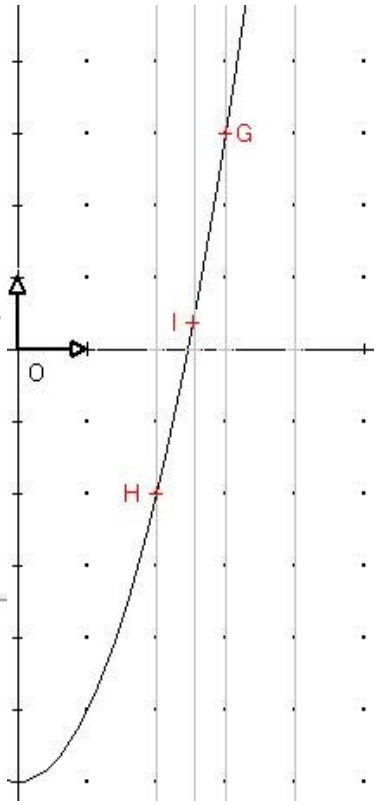
Exercice 2 :
Afficher l'intervalle dans lequel se trouve le nombre pour faciliter la tache à l'utilisateur !
Par exemple : après avoir proposé le nombre 50 l'utilisateur verra s'afficher un message du genre :
C'est moins, le nombre est situé entre 0 et 50
puis après avoir proposé 20
C'est plus, le nombre est situé entre 20 et 50
etc

```
VARIABLES
    hasard EST_DU_TYPE NOMBRE
    proposition EST_DU_TYPE NOMBRE
    compt EST_DU_TYPE NOMBRE
    min EST_DU_TYPE NOMBRE
    max EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    compt PREND_LA_VALEUR 0
    min PREND_LA_VALEUR 0
    max PREND_LA_VALEUR 100
    AFFICHER "un nombre entre "
    AFFICHER min
    AFFICHER " et "
    AFFICHER max
    AFFICHER " VSP "
    hasard PREND_LA_VALEUR floor(random()*100+1)
    proposition PREND_LA_VALEUR 0
    TANT_QUE (proposition!=hasard) FAIRE
        DEBUT_TANT_QUE
        compt PREND_LA_VALEUR compt+1
        LIRE proposition
        SI (proposition<hasard) ALORS
            DEBUT_SI
            min PREND_LA_VALEUR proposition
            AFFICHER "c'est plus , c'est entre "
            AFFICHER min
            AFFICHER " et "
            AFFICHER max
            FIN_SI
        SINON
            DEBUT_SINON
            SI (proposition>hasard) ALORS
                DEBUT_SI
                max PREND_LA_VALEUR proposition
                AFFICHER "c'est moins , c'est entre "
                AFFICHER min
                AFFICHER " et "
                AFFICHER max
                FIN_SI
            FIN_SINON
        FIN_TANT_QUE
    AFFICHER "BRAVO ! vous avez réussi en "
    AFFICHER compt
    AFFICHER " coups."
FIN_ALGORITHME
```

```
            FIN_SI
        SINON
            DEBUT_SINON
            SI (proposition>hasard) ALORS
                DEBUT_SI
                max PREND_LA_VALEUR proposition
                AFFICHER "c'est moins , c'est entre "
                AFFICHER min
                AFFICHER " et "
                AFFICHER max
                FIN_SI
            FIN_SINON
        FIN_TANT_QUE
    AFFICHER "BRAVO ! vous avez réussi en "
    AFFICHER compt
    AFFICHER " coups."
FIN_ALGORITHME
```

VI- Dans la foulée la dichotomie

Il s'agit de calculer la valeur (*approchée souvent !*) qui annule une fonction donnée F1(x).
La méthode est simple : on se place sur un *intervalle de continuité monotone* et on calcule l'image du milieu de cet intervalle, en fonction du signe de cette image ce milieu remplace une des bornes et ainsi de suite.
Exemple : nous considérons la fonction F1(x) = x² – 6 sur le graphique ci-contre sur l'intervalle [0;4]. Nous remarquons que le milieu est 2 et que F1(2) est négatif donc notre intervalle de travail va devenir [2:4], Puis remarquons que F1[3] est positif, notre intervalle devient donc [2,3] etc.




```
VARIABLES
    a EST_DU_TYPE NOMBRE
    min EST_DU_TYPE NOMBRE
    max EST_DU_TYPE NOMBRE
    b EST_DU_TYPE NOMBRE
    middle EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    min PREND_LA_VALEUR 0
    max PREND_LA_VALEUR 5
    a PREND_LA_VALEUR min
    b PREND_LA_VALEUR max
    //on défini une précision de 0.01
    TANT_QUE (abs(a-b)>0.01) FAIRE
        DEBUT_TANT_QUE
        //on met dans middle le point central de l'intervalle
        middle PREND_LA_VALEUR (b+a)/2
        TRACER_POINT (middle,F1(middle))
        //si l'image de middle c'est middle qui devient la borne supérieure de l'intervalle
        //sinon middle  devient la borne inférieure de l'intervalle
        SI (F1(middle)>0) ALORS
            DEBUT_SI
            b PREND_LA_VALEUR middle
            FIN_SI
            SINON
                DEBUT_SINON
                a PREND_LA_VALEUR middle
                FIN_SINON
        FIN_TANT_QUE
    AFFICHER a
    AFFICHER " <solution< "
    AFFICHER b
FIN_ALGORITHME
```

Voici l'algorithme.

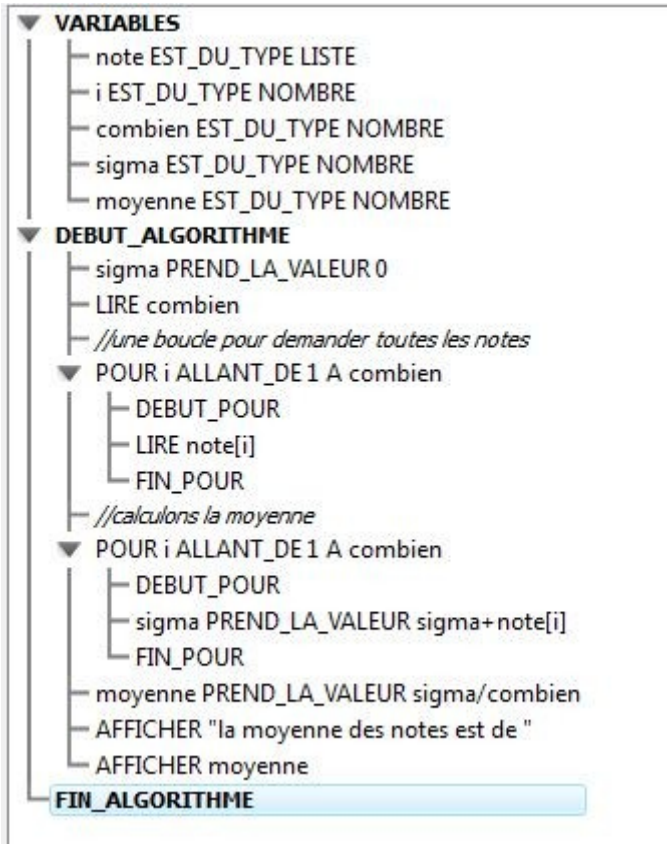
Attention, il faut définir une fonction en utilisant l'onglet **Utiliser une fonction numérique**.
Cette fonction aura pour nom prédéfini **F1(x)**.
Dans l'exemple, si vous désirez utiliser la fonction ci-dessus **x² – 6** il faut taper **pow(x,2) – 6** ou bien tout simplement **x*x – 6**.
Vous pouvez aussi utiliser l'onglet **Dessiner dans un repère** pour voir le parcours fait par le point de recherche **F1(middle).**

Exercice :
Demander à l'utilisateur les deux valeurs de min et de max. Mais il n'est pas possible de demander à l'utilisateur de définir lui-même la fonction F1(x).

VII – Utiliser une liste : moyenne de notes

Pour calculer la moyenne d'un certain nombre de notes (**combien** par exemple) nous devons utiliser *une liste* (mathématiquement on dirait plutôt *un vecteur*),
note[1] contient la première note, **puis note[2]** la seconde etc,
Nous pouvons ainsi faire saisir les notes au sein d'une boucle de longueur **combien**.
Il ne reste qu'à faire la somme
des notes et diviser par
combien.

Voir l'algorithme ci-contre

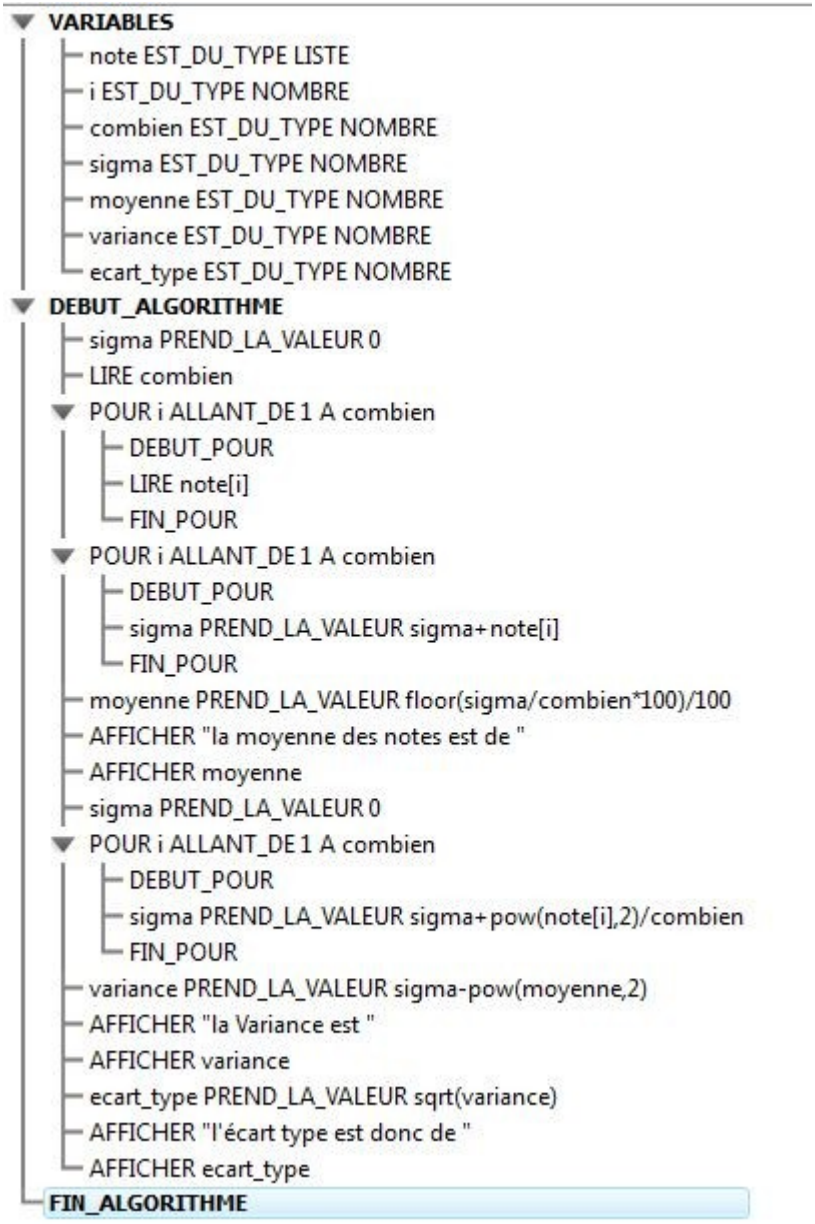


Faites quelques recherches dans votre livre de maths pour récupérer la formule permettant de calculer la Variance d'une série.

Exercice : Faire afficher la moyenne, la variance et l'écart-type de la série de note.

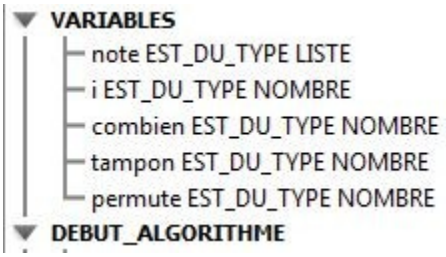
Solution

Exercice : Faire afficher la moyenne, la variance et l'écart-type de la série de note.

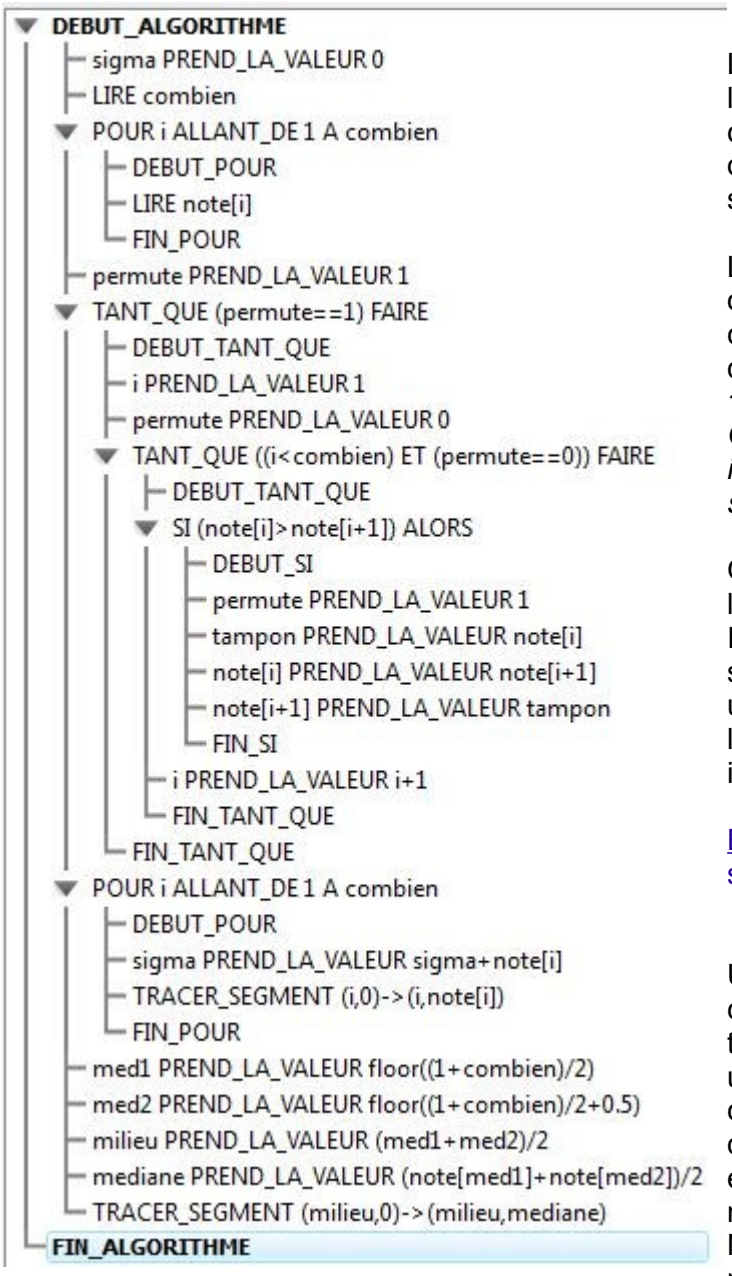


Continuons les statistiques : détermination de la médiane de la série de notes

Pour déterminer la médiane il faut trier la série ! Opération peu simple au niveau des algorithmes.
Il existe de nombreuses façon de réaliser un tri. Nous allons choisir la plus simple mais peut-être pas la plus rapide au niveau de très grandes séries statistiques.



Réalisons un tri dans l'ordre croissant de la série de notes ci-dessus.



Le principe est le suivant : on fait parcourir la série et dés que l'on trouve deux notes consécutives qui ne sont pas dans l'ordre on les inverse et on refait le parcours de la série.

La variable **permute** est mise à zéro puis on fait le parcours de la série de 1 à combien-1. Si on trouve note[i]<note[i+1] dans la série on met la variable **permute** à 1 et on inverse les deux notes.
*Quand la variable **permute** restera à zéro il n'y aura plus rien à inverser et donc la série sera dans l'ordre.*

On utilise un repère pour tracer l'histogramme de la série.
Il faut cliquer
sur **dessiner dans un repère** pour pouvoir utiliser la fonction **TRACER_SEGMENT** de la couleur choisie et après avoir définis les intervalles X et Y du repère,

Exercice : faire un tri décroissant de la série de notes

Une fois la série triée il ne reste plus qu'à calculer la **médiane**. Pour éviter d'avoir à tester la parité du nombre de notes on utilise une petite astuce : on calcule le centre de la série et on fait la moyenne des valeurs entourant ce centre. Si la série est impaire ces deux nombres seront les mêmes sinon ils entoureront le centre.
Nous calculons
med1 = floor((1+combien)/2) puis
med2=floor((1+combien)/2+0.5)
le **milieu** sera **(med1+med2)/2** (*med1 = med2 si on a un nombre impair !*)
mediane sera (note[med1]+note[med2])/2.
Il ne reste qu'à **tracer le médiane** après avoir tracé la série.

Tracé de la médiane de la série de notes
Solution

- VARIABLES
  - note EST_DU_TYPE LISTE
  - i EST_DU_TYPE NOMBRE
  - combien EST_DU_TYPE NOMBRE
  - tampon EST_DU_TYPE NOMBRE
  - permute EST_DU_TYPE NOMBRE
  - med1 EST_DU_TYPE NOMBRE
  - med2 EST_DU_TYPE NOMBRE
  - mediane EST_DU_TYPE NOMBRE
  - milieu EST_DU_TYPE NOMBRE
- DEBUT_ALGORITHME

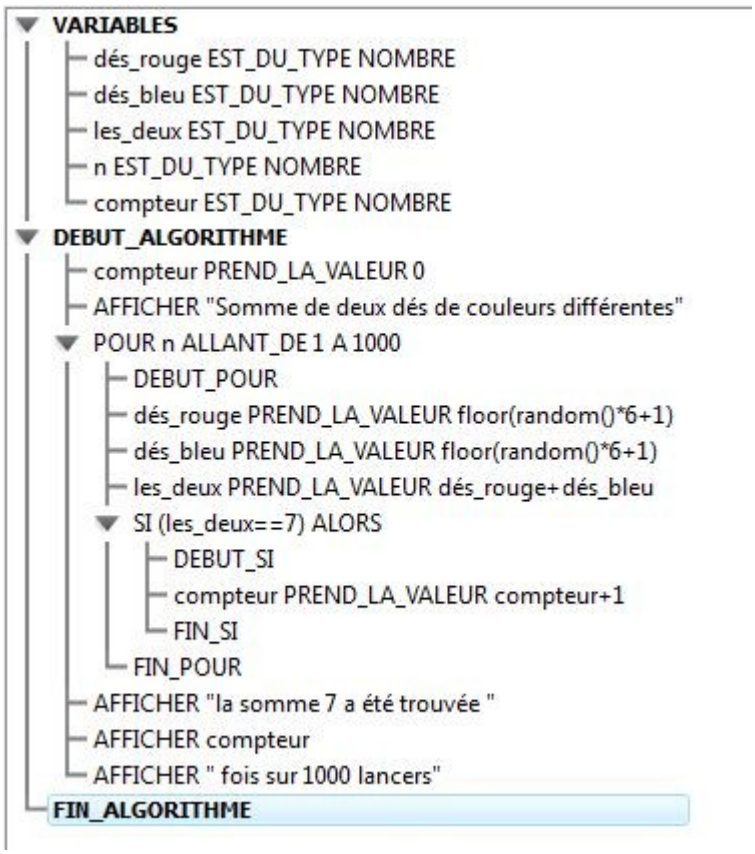
- DEBUT_ALGORITHME
  - sigma PREND_LA_VALEUR 0
  - LIRE combien
  - POUR i ALLANT_DE 1 A combien
    - DEBUT_POUR
    - LIRE note[i]
    - FIN_POUR
  - permute PREND_LA_VALEUR 1
  - TANT_QUE (permute==1) FAIRE
    - DEBUT_TANT_QUE
    - i PREND_LA_VALEUR 1
    - permute PREND_LA_VALEUR 0
    - TANT_QUE ((i<combien) ET (permute==0)) FAIRE
      - DEBUT_TANT_QUE
      - SI (note[i]>note[i+1]) ALORS
        - DEBUT_SI
        - permute PREND_LA_VALEUR 1
        - tampon PREND_LA_VALEUR note[i]
        - note[i] PREND_LA_VALEUR note[i+1]
        - note[i+1] PREND_LA_VALEUR tampon
        - FIN_SI
      - i PREND_LA_VALEUR i+1
      - FIN_TANT_QUE
    - FIN_TANT_QUE
  - POUR i ALLANT_DE 1 A combien
    - DEBUT_POUR
    - sigma PREND_LA_VALEUR sigma+note[i]
    - TRACER_SEGMENT (i,0)->(i,note[i])
    - FIN_POUR
  - med1 PREND_LA_VALEUR floor((1+combien)/2)
  - med2 PREND_LA_VALEUR floor((1+combien)/2+0.5)
  - milieu PREND_LA_VALEUR (med1+med2)/2
  - mediane PREND_LA_VALEUR (note[med1]+note[med2])/2
  - TRACER_SEGMENT (milieu,0)->(milieu,mediane)
- FIN_ALGORITHME

VIII – Un peu de probabilités : la somme de deux dés

Nous savons que la **probabilité** d'apparition de la somme **7** lors du jet de deux dés de couleurs différentes est de **1/6** c'est à dire à peu près **17%** !

Nous pouvons utiliser un algorithme pour vérifier si ce nombre théorique se retrouve sur ***1000 lancers aléatoires*** de la machine.
Utilisons comme nous l'avons déjà fait la fonction **random()** qui génère un décimal entre 0 et 0,99999999
Fabriquons la formule:
random()*6 donne un nombre entre **0 et 5,999999**
random()*6+1 est donc entre **1 et 6,99999**
et il suffit d'utiliser la fonction partie entière **floor()** pour achever la formule
floor(random()*6+1)

Ce qui peut donner un algorithme tel celui-ci



Exercice 1 : demander à l'utilisateur quelle somme il veut tester (entre 2 et 12 bien sur!) et sur combien de lancers il veut faire le test.

*Exercice 2 : demander le nombre de lancers désirés et faire afficher toutes les sommes de 2 à 12, on utilisera **une liste** bien entendu.*

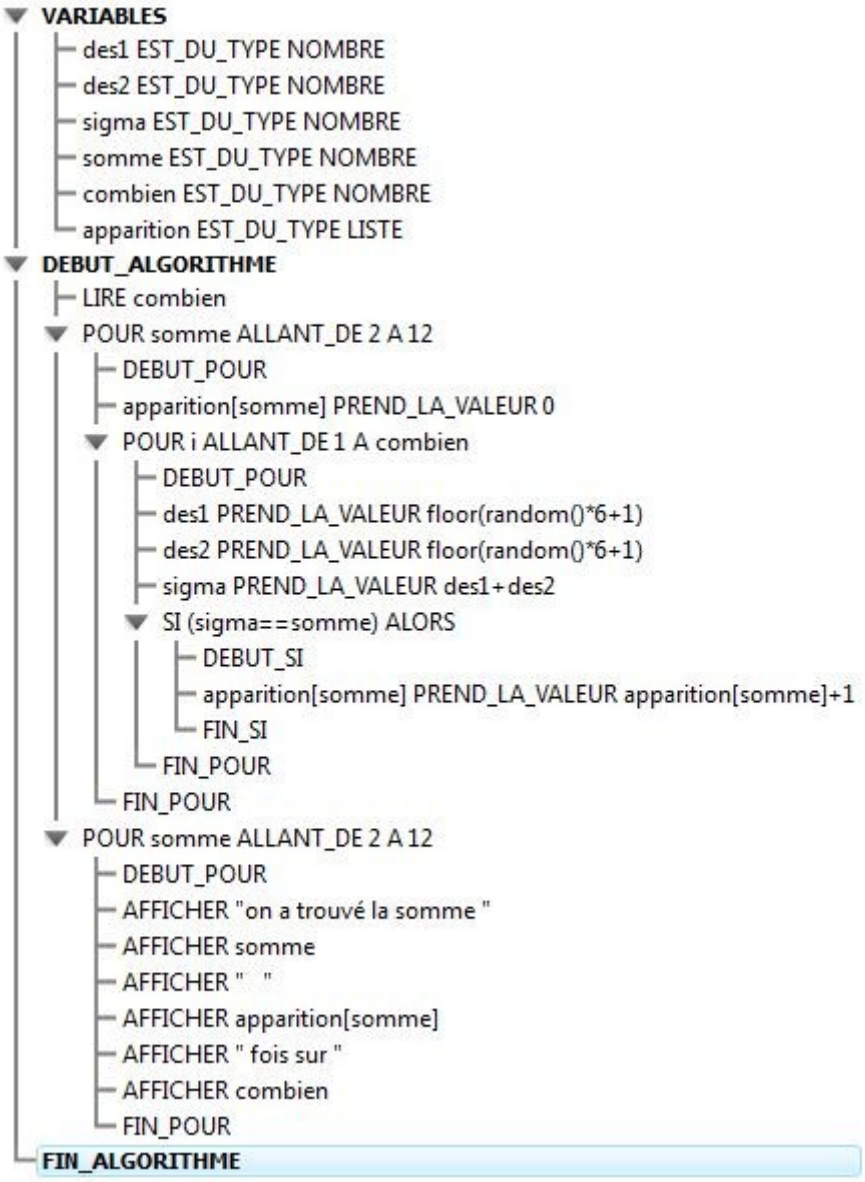
Exercice 1 : demander à l'utilisateur quelle somme il veut tester (entre 2 et 12 bien sur!) et sur combien de lancers il veut faire le test.

Voici une solution

```
VARIABLES
    des1 EST_DU_TYPE NOMBRE
    des2 EST_DU_TYPE NOMBRE
    sigma EST_DU_TYPE NOMBRE
    combien EST_DU_TYPE NOMBRE
    i EST_DU_TYPE NOMBRE
    compteur EST_DU_TYPE NOMBRE
    quelle_somme EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE combien
    LIRE quelle_somme
    //bien penser à mettre le compteur à zéro
    compteur PREND_LA_VALEUR 0
    //on réalise combien fois le tirage des deux dés
    POUR i ALLANT_DE 1 A combien
        DEBUT_POUR
        des1 PREND_LA_VALEUR floor(random()*6+1)
        des2 PREND_LA_VALEUR floor(random()*6+1)
        sigma PREND_LA_VALEUR des1+des2
        //on incrémente compteur uniquement si la somme des dés est 7
        SI (sigma==quelle_somme) ALORS
            DEBUT_SI
            compteur PREND_LA_VALEUR compteur+1
            FIN_SI
        FIN_POUR
    //on réalise l'affichage des résultats
    AFFICHER "on a trouvé la somme "
    AFFICHER quelle_somme
    AFFICHER "  "
    AFFICHER compteur
    AFFICHER " fois sur "
    AFFICHER combien
FIN_ALGORITHME
```

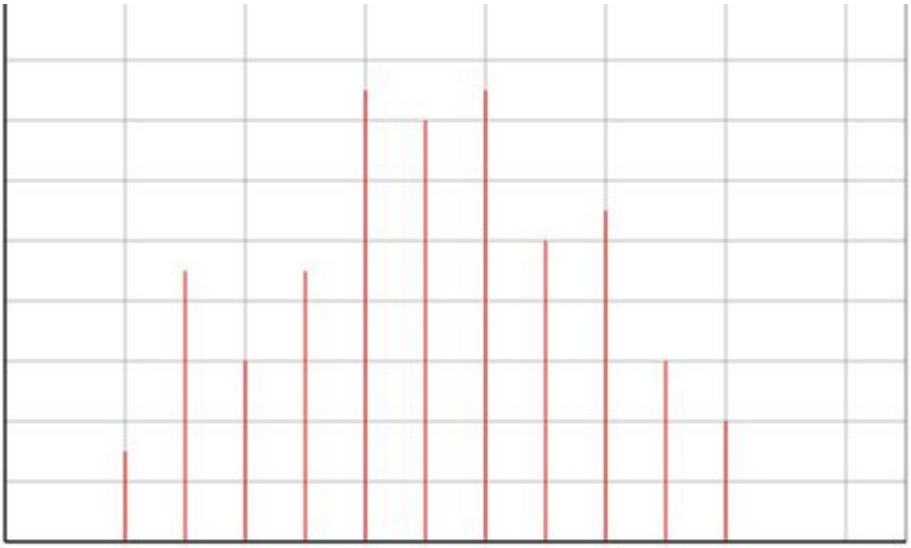
*Exercice 2 : demander le nombre de lancers désirés et faire afficher toutes les sommes de 2 à 12, on utilisera **une liste** bien entendu.*

En voici l'algorithme.

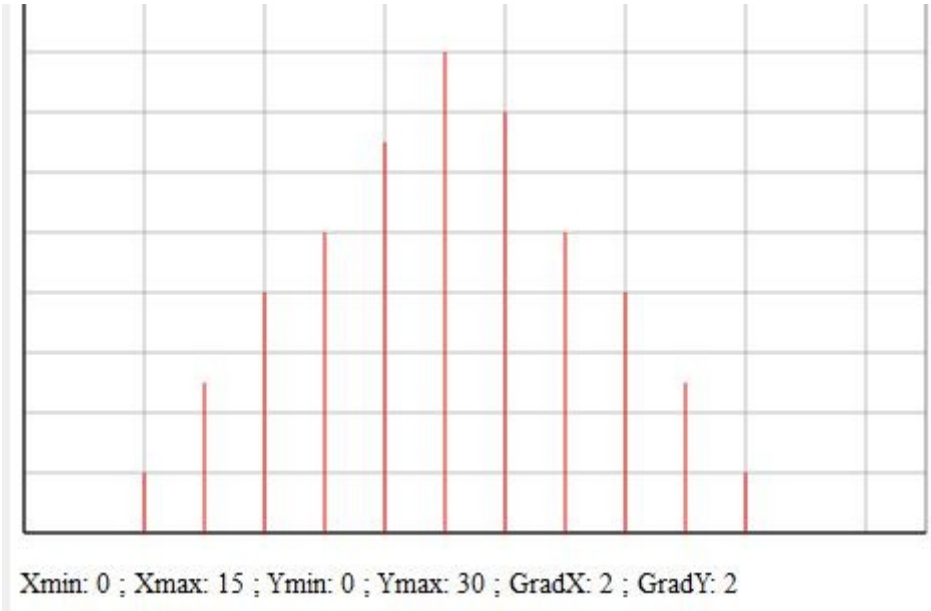


*Exercice 3 : en utilisant la partie **dessiner dans un repère** vous pouvez, à partir de l'algorithme ci-dessus, faire un histogramme des pourcentages*
Il est intéressant d'étudier la régularité et symétrie de l'histogramme en fonction du nombre de lancers demandés.

Voici le graphe pour 100 lancers



Et voici celui pour 10 000 lancers



*Exercice 3 : en utilisant la partie **dessiner dans un repère** vous pouvez, à partir de l'algorithme ci-dessus, faire un histogramme des pourcentages*

En voici la solution