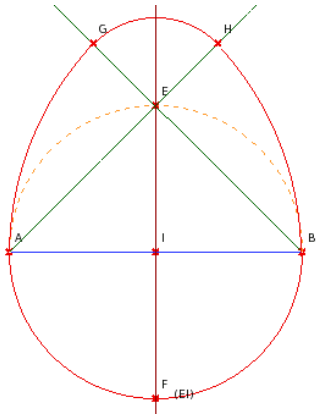
La notion d'algorithme est indépendante de l'informatique. Un algorithme est une suite de processus élémentaires amenant à la résolution d'un problème.

- suivre cet algorithme pour construire un œuf (à la main ou avec Géogébra) :

1. Tracer un segment [AB].
2. Placer son milieu I.
3. Tracer le cercle C de diamètre [AB].
4. Tracer la médiatrice du segment [AB]. Elle coupe le cercle C aux points E et F.
5. Tracer les demi-droites [AE) et [BE).
6. Tracer l'arc de cercle de centre A, de rayon [AB] et d'origine B. Il coupe la demi-droite [AE) au point H.
7. Tracer l'arc de cercle de centre B, de rayon [BA] et d'origine A. Il coupe la demi-droite [BE) au point G.
8. Tracer le quart de cercle de centre E, de rayon [EG] et limité par les points G et H.



- suivre cet algorithme pour calculer rapidement le jour de la semaine de n'importe quel jour des 20ème et 21ème siècles :

D'abord, vous devez mémoriser la table suivante:

Janvier : 1 (0 les années bissextiles)
Février : 4 (3 les années bissextiles)
Mars : 4
Avril :0
Mai : 2
Juin : 5
Juillet : 0
Août : 3
Septembre : 6
Octobre : 1
Novembre : 4
Décembre : 6
Pour retenir plus facilement ces valeurs, groupez-les par 3 : 1 4 4 (carré de 12), 0 2 5 (carré de 5), 0 3 6 (carré de 6), 1 4 6 (carré de 12 + 2).

Prenons une date au hasard, par exemple le 13 novembre 1981.
1) Prenez les deux derniers chiffres de l'année (81 dans notre exemple) et calculez le quotient de la division de ce nombre par 4 (on laisse tomber le reste). 81/4= 20
2) Ajoutez à ce quotient (20) le nombre de départ (81) + le code du mois (novembre = 4) + la date du mois (13)
20+81+4+13 = 118

3) Calculez le reste de la division de ce nombre par 7
118= 16 X 7 + 6

4) Si l'année est « 190 ? », le reste (6) est le numéro du jour de la semaine cherché, sachant que :
Dimanche = 1
Lundi = 2
Mardi = 3
Mercredi = 4
Jeudi = 5
Vendredi = 6
Samedi = 0
Si l'année est « 200 ? », retranchez 1 au reste.

Le 13 novembre 1981 tombait donc un vendredi !

Dans ce stage, nous allons d'abord nous placer dans les objectifs du nouveau programme de 2nde : initier les élèves, en vue de la résolution d'un problème, à mettre en place un algorithme, le décrire clairement, réaliser le programme correspondant, se familiariser avec les instructions élémentaires (entrées, sorties, structures conditionnelles et alternatives). La détermination des processus élémentaires est une activité intéressante ; l'exécution, de peu d'intérêt, est laissée à la machine. Des « gammes » sont nécessaires pour acquérir un peu d'habileté en écriture d'algorithmes.

En vue de faire faire à un ordinateur une série de tâches (afin d'obtenir un résultat attendu), le langage utilisé est **AlgoBox** (logiciel libre et gratuit conçu par Pascal Brachet ; il permet une écriture des programmes très proche de l'écriture algorithmique traditionnelle). Les tests (qui ne sont pas des preuves , mais qui permettent de détecter des erreurs grossières) peuvent être exécutés immédiatement. La programmation n'est que la

réécriture de l'algorithme dans un langage intermédiaire permettant l'exécution par la machine (le microprocesseur de l'ordinateur ne comprend réellement qu'un langage-machine).
Les premiers exemples (certains de P. Brachet) sont utilisables en Seconde.

Un prolongement de cette initiation dans la suite du cursus scolaire étant prévisible, des algorithmes plus complexes sont proposés. Ils révèlent aussi la nécessité de posséder des langages de programmation plus complets (procédures , variables locales, tableaux à plusieurs dimensions, gestion de fichiers, récursivité, etc…).

Un mathématicien doit jongler entre une vision « statique » ($a\mathbb{Z}+b\mathbb{Z}=d\mathbb{Z}$ donne immédiatement le théorème de Bezout) et une vision « dynamique » (calcul effectif du pgcd et des nombres de Bezout).

Problème 1 et vocabulaire : on désire que l'ordinateur convertisse en euros un prix en francs CFP donné par l'utilisateur.

On va donner à la machine un prix en francs, l'ordinateur va traiter ce nombre puis afficher le prix converti en euros. Algorithme élémentaire, où l'on retrouve une entrée, un traitement de données et une sortie d'un résultat.

Ce qui peut s'écrire
- lire le prix en francs
- calculer le prix en euros
- afficher le prix en euros

Il va falloir stocker quelque part le prix en francs (entré par l'utilisateur) et le résultat (le prix en euros). Pour cela on utilise des **variables** où seront stockés des nombres et nous les appelleront *prix_en_francs* et *prix_en_euros* (il est préférable de choisir des noms « parlants »).

Mettre une valeur dans une variable est l'**affectation**.
Conventionnellement, cette affectation est notée : $Angle \leftarrow 41$ qui signifie que la valeur 41 est stockée dans la variable appelée « Angle ». Dans la littérature ou certains langages de programmation on rencontre $Angle := 41$, plus grave $Angle = 41$ ou même sur les calculatrices l'horrible $41 \rightarrow A$.
En AlgoBox, *Angle PREND_LA_VALEUR* 41 est fort convenable.

Première étape : la **déclaration des variables**

Avec **AlgoBox**:

- Cliquer sur le bouton Déclarer nouvelle variable .
- Dans le champ Nom de la variable, entrer *prix_en_francs*, vérifier que le Type de la variable est bien sur NOMBRE et cliquer sur OK .
- Cliquer de nouveau sur le bouton Déclarer nouvelle variable .
- Dans le champ Nom de la variable, entrer *prix_en_euros*, vérifier que le Type de la variable est bien sur NOMBRE et cliquer sur OK .

Deuxième étape : **l'entrée des données** ; il faut permettre à l'utilisateur d'indiquer le prix en francs qu'il veut convertir et stocker ce nombre dans la variable *prix_en_francs*.

- Avec la souris, se placer sur la ligne DEBUT_ALGORITHME, puis cliquer sur le bouton Nouvelle Ligne.
- Cliquer alors sur le bouton Ajouter LIRE variable.

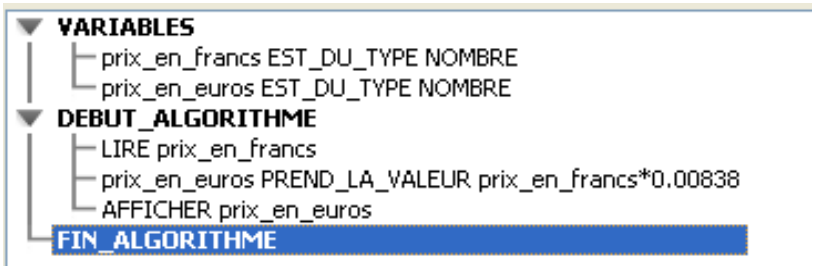
- Dans le champ LIRE la variable, vérifier que *prix_en_francs* est bien sélectionné et cliquer sur OK.

Troisième étape : le **traitement** qui est ici le calcul du prix en euros (1 CFP représente 0,00838 euros).

- Avec la souris, se placer sur la ligne LIRE *prix_en_francs*, puis cliquer sur le bouton Nouvelle Ligne.
- Cliquer alors sur le bouton AFFECTER valeur à variable.
- Sélectionner *prix_en_euros* juste après La variable, taper *prix_en_francs* *0.00838 dans le champ prend la valeur et cliquer sur OK (Attention, le séparateur décimal est le point et pas la virgule).

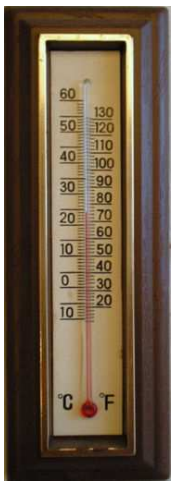
Troisième étape : l'**affichage** du résultat.

- Avec la souris, se placer sur la ligne *prix_en_euros* PREND_LA_VALEUR *prix_en_francs* *0.00838, puis cliquer sur le bouton Nouvelle Ligne.
- Cliquer alors sur le bouton Ajouter AFFICHER variable.
- Sélectionner *prix_en_euros* juste après AFFICHER la variable et cliquer sur OK.



Il n'y a plus qu'à le tester :

- Cliquer sur le bouton Tester Algorithme.
- Dans la fenêtre qui s'affiche, cliquer alors sur le bouton Lancer algorithme.
- Entrer alors un prix à convertir (1000 par exemple) et cliquer sur OK. Le prix en euros apparaît alors dans le cadre Résultat .
- Pour convertir un autre montant, il suffit de cliquer à nouveau sur le bouton Lancer algorithme.
- Une fois les tests terminés, on revient à la fenêtre principale du programme en cliquant sur le bouton fermer en bas de la fenêtre. On peut alors sauvegarder son algorithme en cliquant sur le bouton Sauver.



Attention : LIRE et ECRIRE sont des instructions qu'il faut comprendre « du point de vue du système ». Ainsi il devient compréhensible que LIRE est une demande auprès d'un capteur, d'un appareil photo mais aussi… d'un clavier. De même ECRIRE est une commande pour une machine outil, un disque dur mais aussi …. un écran.

Exemple 2 : écrire un programme **AlgoBox** qui permet de calculer l'aire d'un triangle quand on fournit une base et la hauteur correspondante en cm.

Exemple 3 : écrire un programme **AlgoBox** qui permet de résoudre une équation du premier degré.

Problème 4 : convertir les degrés Fahrenheit en degré Celsius (l'eau gèle à 32°F et bout à 212°F).

Exemple 5 : reprendre avec **AlgoBox** l'exemple du précédent stage "Faites prendre la moitié du nombre pair pensé, puis ajouter 1 et multiplier le résultat par 6; demandez le quotient par 3 du résultat obtenu. Ce quotient diminué de 2 est le nombre pensé."

(Emile Fourrey, Récréations mathématiques, 1905)

*Essayer de bien comprendre les lignes de programme de la forme « nombre PREND_LA_VALEUR nombre * 6 » .*

En effet, au moment de l'exécution, l'ordinateur utilise la valeur connue de nombre, la multiplie par 6 et enfin la stocke à nouveau dans nombre à la place de l'ancienne valeur.

Exemple 6, sujet de brevet, Nouvelle Calédonie 2009 :

Ecrire un programme en **AlgoBox** afin de réaliser des tests.

On considère le programme de calcul ci-dessous.

Programme de calcul :
- **Choisir un nombre de départ**
- **Ajouter 1**
- **Calculer le carré du résultat obtenu**
- **Lui soustraire le carré du nombre de départ**
- **Écrire le résultat final**

1. a) Vérifier que lorsque le nombre de départ est 1, on obtient 3 au résultat final.
   b) Lorsque le nombre de départ est 2, quel résultat final obtient-on ?
   c) Le nombre de départ étant x, exprimer le résultat final en fonction de x.

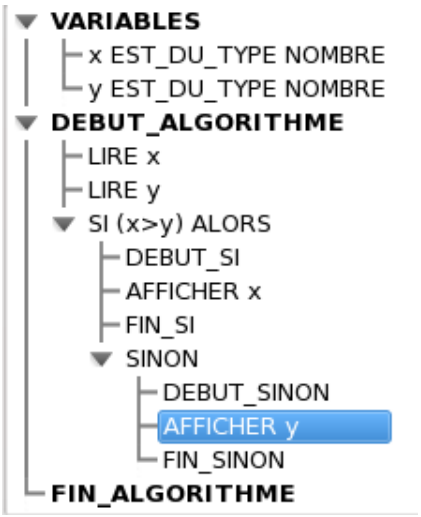
2. On considère l'expression $P = (x+1)^2 - x^2$. Développer puis réduire l'expression P.

3. Quel nombre de départ doit-on choisir pour obtenir un résultat final égal à 15 ?

Exemple 7, l'instruction conditionnelle : concevoir un algorithme qui demande à l'utilisateur d'entrer deux nombres (stockés dans les variables x et y) et qui affiche le plus grand des deux.

Analyse du problème :

- Il va falloir d'abord déclarer les deux variables x et y du type NOMBRE, puis LIRE le contenu de ces deux variables .
- On utilise ensuite le raisonnement suivant :
  - SI x>y alors le plus grand des deux nombres correspond à la valeur de x qu'on affiche.
  - SINON (dans la cas contraire), le plus grand des deux nombres correspond à la valeur de y qu'on affiche.



Avec **AlgoBox**: on crée une nouvelle ligne (bouton Nouvelle Ligne) après les lignes LIRE x et LIRE y, puis on clique sur le bouton Ajouter SI...ALORS.

  - Dans le champ après SI la condition de la boîte de dialogue, on entre x>y et on coche la case Ajouter SINON avant de cliquer sur OK.
  - On se place sur la ligne vide entre DEBUT_SI et FIN_SI, puis on clique sur Ajouter AFFICHER variable et on sélectionne x comme variable à afficher.

- On se place sur la ligne vide entre DEBUT_SINON et FIN_SINON, puis on clique sur Ajouter AFFICHER variable et on sélectionne y comme variable à afficher.

Problème 8 : écrire un programme qui permet de sortir le plus grand de trois nombres x, y et z.

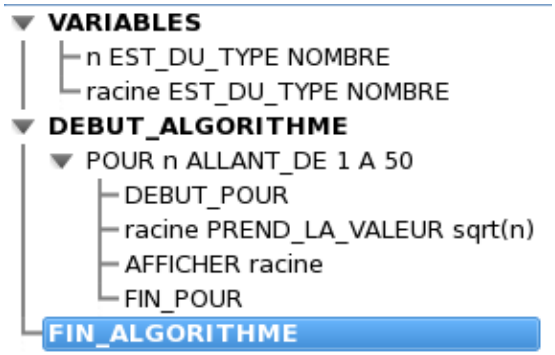
Problème 9 : donner la valeur absolue d'un nombre.

Exemple 10 : écrire un programme **AlgoBox** qui permet de résoudre une équation du premier degré.

Problème 11 : déterminer si un triangle est isocèle.

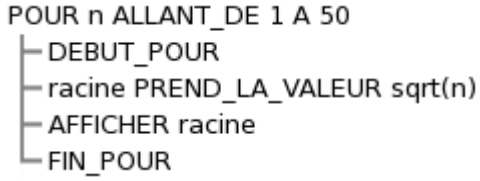
```
début
        donner à i la valeur 1
        répéter 3 fois
                lire la valeur de abs[i]                    % liste des abscisses %
                lire la valeur de ord[i]                    % liste des ordonnées %
                donner à i la valeur i + 1
        fin
        donner à M1M2 la valeur (abs[2] - abs[1])² + (ord[2] - ord[1])²
        dnner à M1M3 la valeur (abs[3] - abs[1])² + (ord[3] - ord[1])²
        si M1M2 = M1M3 alors
                afficher "C'est un triangle isocèle en M1"
        sinon
                afficher "Ce n'est pas un triangle isocèle en M1"
        fin
fin
```

Exemple 12, la répétition : que réalise ce programme ?



On souhaitait réaliser l'affichage des racines carrées des entiers de 1 à 50.

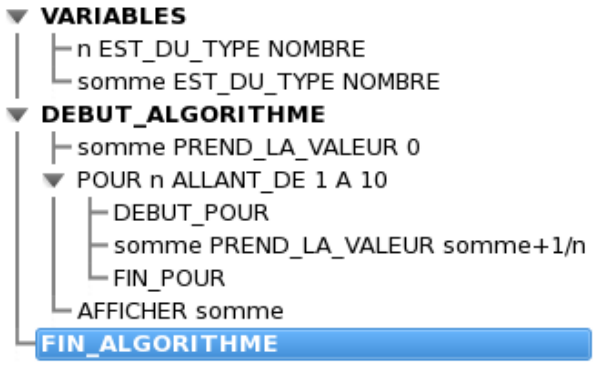
- Il nous a fallu une nouvelle variable n qui va servir à représenter tous les entiers de 1 à 50 (n est appelé **compteur** de la boucle).
- Regardons maintenant le code de la boucle:



  - Première étape : l'ordinateur affecte à la variable n la valeur 1. Puis il effectue les opérations comprises entre DEBUT_POUR et FIN_POUR en remplaçant n par 1. Autrement dit, racine va prendre la valeur sqrt(1) et l'ordinateur affiche le résultat.
  - Deuxième étape : l'ordinateur augmente automatiquement de 1 la valeur de n qui vaut donc maintenant 2. Puis il effectue à nouveau les opérations comprises entre DEBUT_POUR et FIN_POUR en remplaçant cette fois-ci n par 2. Autrement dit, racine va prendre la valeur sqrt(2) et l'ordinateur affiche le résultat.
  - Troisième étape : l'ordinateur augmente de nouveau automatiquement de 1 la valeur de n qui vaut donc maintenant 3. Puis il effectue les opérations comprises entre DEBUT_POUR et FIN_POUR en remplaçant n par 3. Autrement dit, racine va prendre la valeur sqrt(3) et l'ordinateur affiche le résultat.

- L'ordinateur continue ainsi le processus jusqu'à ce qu'il ait traité le cas où n vaut 50. On obtient bien ainsi la liste des racines carrées des entiers de 1 jusqu'à 50.

Exemple 13 : A quel calcul mathématique correspond la valeur de la variable *somme* qui est affichée à la fin de l'exécution de l'algorithme?



Exemple 14 : Ecrire un programme de calcul de factorielle.

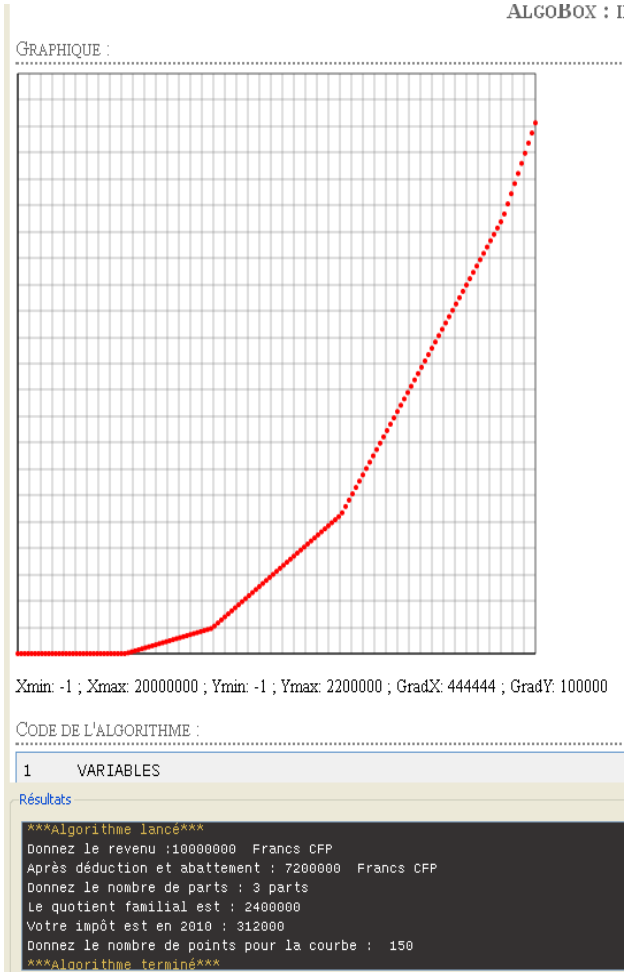
Exemple 15 : Calcul du n° terme d'une suite récurrente.

Exemple 16 : Faire afficher la fonction partie entière.

Exemple 17 : écrire un programme qui permet de calculer l'impôt sur le revenu et tracer la fonction « par morceaux » liée à ce calcul.

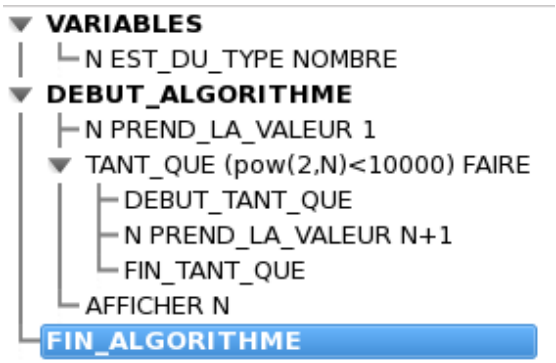
```
1   VARIABLES
2    R EST_DU_TYPE NOMBRE
3    N EST_DU_TYPE NOMBRE
4    QF EST_DU_TYPE NOMBRE
5    impot EST_DU_TYPE NOMBRE
6    xmin EST_DU_TYPE NOMBRE
7    xmax EST_DU_TYPE NOMBRE
8    ymin EST_DU_TYPE NOMBRE
9    ymax EST_DU_TYPE NOMBRE
10   nombre_points EST_DU_TYPE NOMBRE
11   i EST_DU_TYPE NOMBRE
12   pas EST_DU_TYPE NOMBRE
13  DEBUT_ALGORITHME
14   AFFICHER "Donnez le revenu :"
15   LIRE R
16   AFFICHER R
17   AFFICHER " Francs CFP"
18   R PREND_LA_VALEUR R*0.9*0.8
19   AFFICHER "Après déduction et abattement : "
20   AFFICHER R
21   AFFICHER " Francs CFP"
22   AFFICHER "Donnez le nombre de parts : "
23   LIRE N
24   AFFICHER N
25   AFFICHER " parts"
26   QF PREND_LA_VALEUR R/N
27   AFFICHER "Le quotient familial est : "
28   AFFICHER QF
29   SI (QF<=1000000) ALORS
30    DEBUT_SI
31    impot PREND_LA_VALEUR 0
32    FIN_SI
33   SI (QF > 1000000 ET QF<= 1800000) ALORS
34    DEBUT_SI
35    impot PREND_LA_VALEUR R*0.04-40000*N
36    FIN_SI
37   SI (QF>1800000 ET QF <=3000000) ALORS
38    DEBUT_SI
39    impot PREND_LA_VALEUR R*0.12-184000*N
40    FIN_SI
41   SI (QF>3000000 ET QF <=4500000) ALORS
42    DEBUT_SI
43    impot PREND_LA_VALEUR R*0.25-574000*N
44    FIN_SI
45   SI (QF>4500000) ALORS
46    DEBUT_SI
47    impot PREND_LA_VALEUR R*0.4-1249000*N
48    FIN_SI
49   AFFICHER "Votre impôt est en 2010 : "
50   AFFICHER impot
51   AFFICHER "Donnez le nombre de points pour la
courbe : "
52   LIRE nombre_points
53   xmin PREND_LA_VALEUR 0
54   AFFICHER nombre_points
55   ymin PREND_LA_VALEUR 0
56   xmax PREND_LA_VALEUR 20000000
57   pas PREND_LA_VALEUR xmax/nombre_points
58   ymax PREND_LA_VALEUR xmax*0.9*0.8*0.4-
1249000*N
59   POUR i ALLANT_DE 0 A nombre_points
60    DEBUT_POUR
61    R PREND_LA_VALEUR i*pas*0.9*0.8
62    QF PREND_LA_VALEUR R/N
63    SI (QF<=1000000) ALORS
64     DEBUT_SI
```

```
65        impot PREND_LA_VALEUR 0
66        FIN_SI
67      SI (QF > 1000000 ET QF<= 1800000) ALORS
68        DEBUT_SI
69        impot PREND_LA_VALEUR R*0.04-40000*N
70        FIN_SI
71      SI (QF>1800000 ET QF <=3000000) ALORS
72        DEBUT_SI
73        impot PREND_LA_VALEUR R*0.12-184000*N
74        FIN_SI
75      SI (QF>3000000 ET QF <=4500000) ALORS
76        DEBUT_SI
77        impot PREND_LA_VALEUR R*0.25-574000*N
78        FIN_SI
79      SI (QF>4500000) ALORS
80        DEBUT_SI
81        impot PREND_LA_VALEUR R*0.4-1249000*N
82        FIN_SI
83      TRACER_POINT (i*pas,impot)
84      FIN_POUR
85    FIN_ALGORITHME
```



Exemple 18, autre type de boucle : dans les exemples précédents, le nombre de passages dans la boucle était déterminé. Il n'en est pas toujours ainsi.

On cherche à connaître le plus petit entier N tel que 2^N soit supérieur ou égal à 10000.
Pour résoudre ce problème de façon algorithmique, l'idée est de calculer les puissances consécutives de 2 jusqu'à ce qu'on atteigne 10000. Une structure TANT QUE est particulièrement adaptée à ce genre de problème car on ne sait pas a priori combien de calculs seront nécessaires.



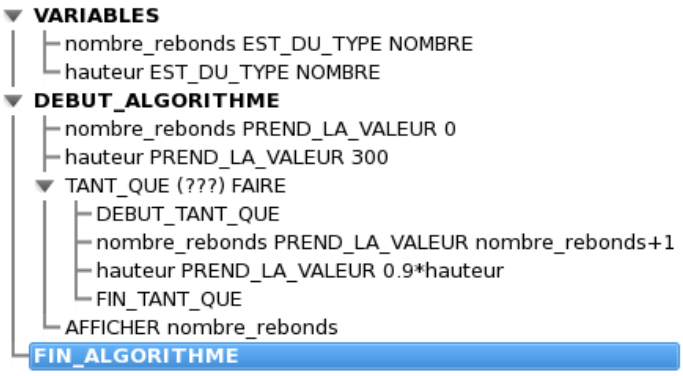
Explication : tant que 2^N<10000 , on augmente de 1 la valeur de N (qui vaut 1 au début de l'algorithme). Par contre, dés que 2^N dépasse 10000, on a atteint notre but et l'instruction entre DEBUT_TANT_QUE et FIN_TANT_QUE n'est pas traitée : on passe alors directement à l'affichage du résultat.

Note : il est indispensable d'affecter à N la valeur 1 avant d'ajouter la structure TANT QUE, sans quoi l'algorithme ne peut plus fonctionner.

Exemple 19 :

- On lance une balle d'une hauteur initiale de 300 cm.
- On suppose qu'à chaque rebond, la balle perd 10% de sa hauteur (la hauteur est donc multipliée par 0.9 à chaque rebond).
- On cherche à savoir le nombre de rebonds nécessaire pour que la hauteur de la balle soit inférieure ou égale à 10 cm.

Par quelle condition faut-il remplacer ??? dans la ligne TANT_QUE (???) FAIRE pour que l'algorithme réponde au problème.



Problème 20 : on souhaite diviser l'entier a par b en n'utilisant que additions et soustractions .

23-7=16 16-7 = 9 9-7=2 le quotient est 3 (on a pu retrancher 3 fois 7 de 23) et le reste est 2

Exemple 21 : Trouver en exécutant pas à pas à la main les valeurs affichées :

```
début
        donner à I la valeur 5
        donner à S la valeur 2
        répéter tant que I <=31
                afficher I
                donner à  I  la valeur  I + S
                donner à  S  la valeur 6 - S
        fin
fin
```

Problème 22 : explorer la « suite de Syracuse »

On part d'un entier A ; si A=1, stop ; si A pair on remplace A par A/2 sinon par 3A+1.

Problème 23 : savoir si un nombre est parfait (6 est « parfait » car 6=1+2+3 , il est égal à la somme de ses diviseurs).

```
VARIABLES
 x EST_DU_TYPE NOMBRE
 t EST_DU_TYPE LISTE
 i EST_DU_TYPE NOMBRE
 diviseur EST_DU_TYPE NOMBRE
 resultat EST_DU_TYPE NOMBRE
 intermediaire EST_DU_TYPE NOMBRE
 Preuve EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
 Preuve PREND_LA_VALEUR 0
 LIRE x
 diviseur PREND_LA_VALEUR 2
 t[0] PREND_LA_VALEUR 1
 i PREND_LA_VALEUR 1
TANT_QUE (diviseur <= x / 2) FAIRE
 DEBUT_TANT_QUE
 resultat PREND_LA_VALEUR x / diviseur
 intermediaire PREND_LA_VALEUR floor (resultat)
 SI (intermediaire == resultat) ALORS
  DEBUT_SI
  t[i] PREND_LA_VALEUR diviseur
  i PREND_LA_VALEUR i + 1
  FIN_SI
 diviseur PREND_LA_VALEUR diviseur + 1
 FIN_TANT_QUE
i PREND_LA_VALEUR i - 1
TANT_QUE (i >= 0 ) FAIRE
 DEBUT_TANT_QUE
```

```
  Preuve PREND_LA_VALEUR Preuve + t[i]
  AFFICHER t[i]
  i PREND_LA_VALEUR i - 1
  FIN_TANT_QUE
AFFICHER "-------------------------------------"
AFFICHER Preuve
SI (x == Preuve) ALORS
  DEBUT_SI
  AFFICHER x
  AFFICHER " est un nombre parfait"
  FIN_SI
  SINON
   DEBUT_SINON
   AFFICHER x
   AFFICHER " n'est pas un nombre parfait"
   FIN_SINON
FIN_ALGORITHME
```

Problème 24 : trouver les nombres parfaits entre 1 et 1000 (1, 6, 28, …)

Problème 25 : savoir si deux nombres sont amis.

220 et 284 sont « amis » car la somme des diviseurs de 220 est égale à 284 et la somme des diviseurs de 284 est égale à 220.

Exemple 26, les chaînes : Algobox connaît les variables de type « chaînes » .

- Le contenu d'une chaîne doit être encadré par des guillemets :

Exemple : a *prend la valeur* "bonjour" (a étant une variable du type chaîne)

- Il est possible d'ajouter (concaténer) des chaînes :

Exemple : b *prend la valeur* a+"bonjour" (a et b étant des variables du type CHAINE)

- Il est possible d'extraire le contenu d'une chaîne avec l'instruction chaine.substr(position_premier_caractère_à_extraire,nombre_de_caractères_à_extraire).

Attention : le premier caractère a pour position 0 (et pas 1)

Exemple : b *prend la valeur* a.substr(4,2) (b sera alors formé des 5ème et 6ème caractères de a ; a et b étant des variables du type CHAINE)

- Un nombre peut-être transformé en chaîne avec l'instruction nombre.toString()

Exemple : machaine *prend la valeur* nb.toString() (nb étant une variable du type NOMBRE et machaine étant une variable du type CHAINE)

- La longueur d'une chaîne peut-être obtenue avec l'instruction chaine.length

Exemple : longueur *prend la valeur* machaine.length (longueur étant une variable du type NOMBRE et machaine étant une variable du type CHAINE)

- L'instruction machaine.charCodeAt(pos) permet d'obtenir le nombre égal au code ascii de la lettre figurant à la position pos dans la chaîne machaine (Attention : le premier caractère a pour position 0).
- Inversement, l'instruction String.fromCharCode(nombre) renvoie une chaîne contenant le caractère dont le code ascii est égal à nombre.

Algorithme inversant une chaîne afin de savoir s'il s'agit d'un palindrome

```
VARIABLES
2     x EST_DU_TYPE NOMBRE
3     palind EST_DU_TYPE CHAINE
4     result EST_DU_TYPE CHAINE
5     longueur EST_DU_TYPE NOMBRE
6   DEBUT_ALGORITHME
7     //Vidons la chaine result
8     result PREND_LA_VALEUR ""
9     LIRE palind
10    //On a besoin de la longueur de la chaine palind
11    longueur PREND_LA_VALEUR palind.length
12    POUR x ALLANT_DE 1 A longueur+1
```

```
13      DEBUT_POUR
14      //Le dernier caractère est mis en premier dans result puis l'avant dernier en second etc
15      result PREND_LA_VALEUR result+palind.substr (longueur -x+1,1)
16      FIN_POUR
17     AFFICHER result
18    FIN_ALGORITHME
```

Exemple 27, tris :

Il s'agit de ranger une collection d'objets. L'ordre sera ordre numérique ou éventuellement l'ordre lexicographique. De très nombreuses méthodes de tris existent ; elles n'ont pas toutes la même rapidité.

Si on demande aux élèves de ranger la liste (7, 32, 1, 5, 49, 122, 8, 3, 51, 13, 17, 28, 4, 99, 18) il y a vite nécessité de dégager une procédure systématique, procédure qu'il faudrait pouvoir « expliquer » à une machine !
La nécessité d'échanger deux nombres apparaît vite aussi. On peut aussi réfléchir à « comment trouver le plus grand nombre de cette liste ? » et à rédiger un tel algorithme.

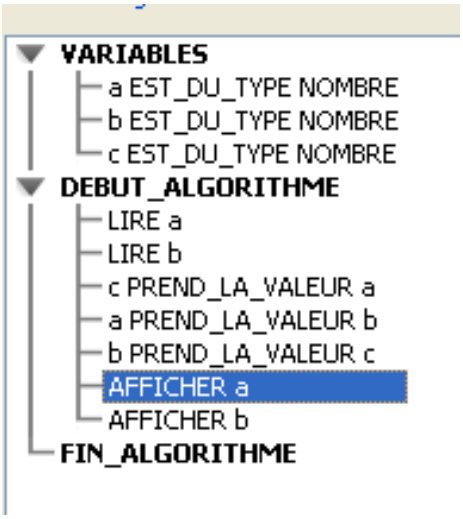
Il existe de très nombreux algorithmes de tris (comptage, transposition..). Il n'ont pas toutes la même efficacité (temps de calculs, mobilisation de la mémoire ..). Il est possible de comparer leurs performances en faisant d'abord remplir aléatoirement un tableau de 200 nombres à trier par diverses méthodes….

A défaut de tableaux, Algobox connaît les variables de type « liste »

- Les listes AlgoBox sont des listes numérotées de nombres.
- Si vous sélectionnez pour *la variable* une variable du type LISTE, vous devez indiquer dans le champ *rang du terme de la liste* le numéro du terme de la liste auquel vous voulez affecter une valeur.
- Pour utiliser un terme d'une liste dans un calcul, il faut utiliser la syntaxe suivante : nomliste[rang]
  Exemple : moy *prend la valeur* (maliste[1]+maliste[2]+maliste[3])/3 (la variable du type NOMBRE moy contiendra la moyenne des trois premiers termes de la liste maliste)

a) Echange

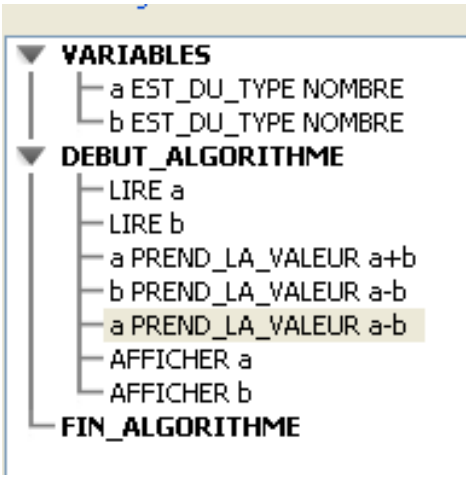
Exercice : donner un algorithme permettant d'échanger deux nombres A et B, c'est-à-dire les entrées sont A et B et les sorties B et A. Posé ainsi, le problème n'est pas toujours ben perçu.



Personnellement, ce jour là j'arrive en classe avec un verre de café et une tasse de chocolat : je pose alors la question, comment échanger les contenants ?

La nécessité d'un 3° espace de stockage est là bien détectée et comprise.

Quoiqu'en informatique il ne s'agit pas de transfert, mais de recopie. Une solution existe sans utiliser de variables supplémentaires.
Etudier ceci …



(les rusés de la programmation connaissent le véloce : a :=a xor b ; b :=b xor a ; a :=a xor b !!!)

Exercice : donner un algorithme permettant d'échanger circulairement trois nombres A, B et C.

b) Le tri par sélection

Le principe est d'aller sélectionner le plus petit de la liste et le placer en tête .

Puis poursuivre avec la liste amputée du premier nombre.

Puis …

```
 VARIABLES
2      i EST_DU_TYPE NOMBRE
3      combien EST_DU_TYPE NOMBRE
4      a EST_DU_TYPE LISTE
5      indice EST_DU_TYPE NOMBRE
6      j EST_DU_TYPE NOMBRE
7      petit EST_DU_TYPE NOMBRE
8      k EST_DU_TYPE NOMBRE
9    DEBUT_ALGORITHME
10     AFFICHER "Combien d'éléments à ranger ?:"
11     LIRE combien
12     AFFICHER combien
13     POUR k ALLANT_DE 1 A combien
14       DEBUT_POUR
15       LIRE a[k]
16       FIN_POUR
17     POUR k ALLANT_DE 1 A combien
18       DEBUT_POUR
19       AFFICHER a[k]
20       FIN_POUR
21     AFFICHER " "
22     POUR i ALLANT_DE 1 A combien - 1
23       DEBUT_POUR
24       indice PREND_LA_VALEUR i
25       petit PREND_LA_VALEUR a[indice]
26       POUR j ALLANT_DE i+1 A combien
27         DEBUT_POUR
28         SI (a[j]<petit) ALORS
29           DEBUT_SI
30           indice PREND_LA_VALEUR j
31           petit PREND_LA_VALEUR a[j]
32           FIN_SI
33         FIN_POUR
34       a[indice] PREND_LA_VALEUR a[i]
35       a[i] PREND_LA_VALEUR petit
36       POUR k ALLANT_DE 1 A combien
37         DEBUT_POUR
38         AFFICHER a[k]
39         FIN_POUR
40       AFFICHER " "
41       FIN_POUR
42   FIN_ALGORITHME
```

c) Le tri par propagation (à bulles)

Le principe est de comparer deux à deux les éléments e1 et e2 consécutifs d'un tableau et d'effecteur une permutation si e1 > e2. On continue de trier jusqu'à ce qu'il n'y ait plus de permutation.
Ainsi, par permutations successives le plus grand élément « remonte » à la fin du tableau ; ce principe sera répété pour chacun des éléments.

Exemple 28, la dichotomie :

On désire obtenir la valeur approchée de la solution de l'équation: $x\sqrt{x+1}-4=0$:

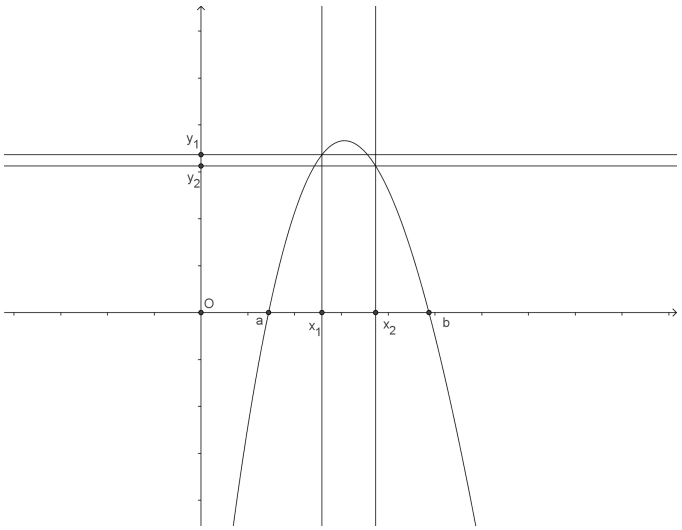
```
1  VARIABLES
2   a EST_DU_TYPE NOMBRE
3   b EST_DU_TYPE NOMBRE
4   ecart EST_DU_TYPE NOMBRE
5   milieu EST_DU_TYPE NOMBRE
6   image EST_DU_TYPE NOMBRE
7  DEBUT_ALGORITHME
8   a PREND_LA_VALEUR 1
9   b PREND_LA_VALEUR 3
10   ecart PREND_LA_VALEUR b-a
11   TANT_QUE (ecart>0.001) FAIRE
12    DEBUT_TANT_QUE
13    milieu PREND_LA_VALEUR (a+b)/2
14    image PREND_LA_VALEUR F1(milieu)
15    AFFICHER "("
16    AFFICHER a
17    AFFICHER ";"
18    AFFICHER b
19    AFFICHER ")"
20    AFFICHER "milieu="
21    AFFICHER milieu
22    AFFICHER ";"
23    AFFICHER "image par la fontion="
24    AFFICHER image
25    SI (F1(milieu)<0) ALORS
26     DEBUT_SI
27     a PREND_LA_VALEUR milieu
28     FIN_SI
29     SINON
30      DEBUT_SINON
31      b PREND_LA_VALEUR milieu
32      FIN_SINON
33    ecart PREND_LA_VALEUR b-a
34    FIN_TANT_QUE
35   AFFICHER "la valeur qui annule la fonction est proche
de :"
36   AFFICHER milieu
37  FIN_ALGORITHME
```

Exemple 29, recherche d'un maximum d'une fonction par trisection :

f est définie sur $[a\ ;b]$.

On note $h=b-a$ et on calcule

$$y_1=f(a+\frac{h}{3}) \qquad y_2=f(a+\frac{2h}{3})=f(b-\frac{h}{3})$$



Si $y_1 \le y_2$, on peut exclure le premier tiers de l'intervalle $[a\ ;b]$ et on recommence en remplaçant a par $a+\frac{h}{3}$. Sinon, on exclut le dernier tiers et on recommence en remplaçant b par $b-\frac{h}{3}$.

L'amplitude h est le terme général d'une suite géométrique de raison positive, inférieure à 1, donc tend vers 0. Le procédé converge vers un intervalle réduit à un point pour lequel f est maximale. Dans la pratique on s'arrête si h est inférieur à une précision fixée par l'utilisateur. Nous choisissons ici $h=10^{-6}$.

Exemple 30, approximation de $\sqrt{2}$:

a) On doit à Théon de Smyrne ces deux suites définies par récurrence.

Etant donné deux entiers naturels non nuls a et b :

A. Calculer $a' = a + 2b$ *et* $b' = a + b$.

B. Remplacer a *par* a' *et* b *par et recommencer le calcul précédent avec les nouvelles valeurs de* a *et de* b .

$\frac{a'}{b'}$ est une meilleure approximation de $\sqrt{2}$ que $\frac{a}{b}$.

Initialisation : on choisit pour a et b deux entiers simples dont le quotient est voisin de $\sqrt{2}$. (exemple : $a = 3$ et $b = 2$) . Le calcul est stoppé dès que les valeurs correspondantes des deux dernières colonnes sont identiques.

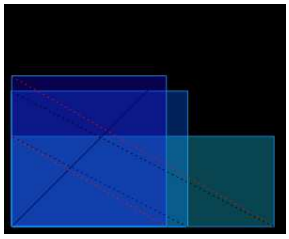
a	b	a'	b'	a/b	a'/b'
3	2	7	5	1,500000000000000000000	1,400000000000000000000
7	5	17	12	1,400000000000000000000	1,416666666666700000000
17	12	41	29	1,416666666666700000000	1,413793103448280000000
41	29	99	70	1,413793103448280000000	1,414285714285710000000
99	70	239	169	1,414285714285710000000	1,414201183431950000000
239	169	577	408	1,414201183431950000000	1,414215686274510000000
577	408	1393	985	1,414215686274510000000	1,414213197969540000000

b) Algorithme de Héron

Chez les mathématiciens grecs, extraire la racine carré de A c'est trouver un carré dont l'aire soit A. En prenant un rectangle de côté arbitraire X et de même aire, il est nécessaire que l'autre côté ait pour longueur A/X. Mais ce rectangle n'est pas carré (en général). Pour le rendre *moins rectangle*, il suffit de prendre un rectangle dont la longueur est la moyenne arithmétique des deux côtés précédents soit

$$\frac{X + A/X}{2}$$

et dont l'aire reste A. En réitérant infiniment le processus, on transforme petit à petit le rectangle en carré de même aire



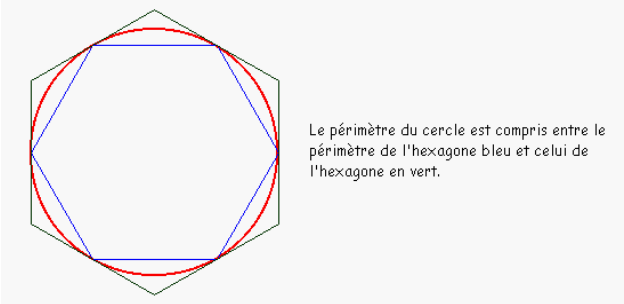
Les rectangles ont même aire, Chaque rectangle a pour longueur la moyenne des dimensions du rectangle précédent.

Alors que la convergence de Théon de Smyrne est linéaire, cette méthode (dite aussi « de Newton ») est quadratique.

Exemple 31, approximations de Pi :

a) Calcul de Π par la méthode des polygones inscrits

La méthode (Archimède, vers 250 avant J-C) consiste à calculer le périmètre du polygone régulier à 2^n côtés inscrit dans le cercle de rayon 1.
Ce périmètre tend vers 2Π.



b) Calcul de Π par une méthode statistique

Le rapport entre l'aire du quart de disque et l'aire du carré est de $\Pi / 4$
On choisit au hasard des points dans le carré et on compte ceux qui sont dans le quart de disque.
On a alors : Π = 4 x (nb de points dans le quart de disque) / (nb de points dans le carré)
Mais la vitesse de convergence n'est pas terrible !


```
1    VARIABLES
2      x EST_DU_TYPE NOMBRE
3      y EST_DU_TYPE NOMBRE
4      r EST_DU_TYPE NOMBRE
5      n EST_DU_TYPE NOMBRE
6      i EST_DU_TYPE NOMBRE
7      nombre_points EST_DU_TYPE NOMBRE
8    DEBUT_ALGORITHME
9      n PREND_LA_VALEUR 0
10     AFFICHER "Nombre de points ? :"
11     LIRE nombre_points
12     AFFICHER nombre_points
13     POUR i ALLANT_DE 1 A nombre_points
14       DEBUT_POUR
15       x PREND_LA_VALEUR random()
16       y PREND_LA_VALEUR random()
17       r PREND_LA_VALEUR sqrt(x*x+y*y)
18       SI (r<1) ALORS
19         DEBUT_SI
20         TRACER_POINT (x,y)
21         n PREND_LA_VALEUR n+1
22         FIN_SI
23         SINON
24           DEBUT_SINON
25           TRACER_POINT (x,y)
26           FIN_SINON
27       FIN_POUR
28     n PREND_LA_VALEUR n*4/nombre_points
29     AFFICHER "Valeur approchée de PI : "
30     AFFICHER n
31   FIN_ALGORITHME
```

c) Calcul de Π par une formule d'Euler (Xcas)

$$2\times \lim_{n\to\infty} \sum_{i=1}^{n} \frac{1.2. \ldots .i}{1.3. \ldots (2.i+1)} = \Pi$$

```
Save                                   = real RAD 40
1 n:=200;
j:=1;
s:=1;
k:=1;
Digits:=40;
while(k<n)
{
s:=s+product(j/(2*j+1),j,1,k,1);
k:=k+1;
};
print(evalf(2*s));
3.1415926535897932384626433832795028841975
Evaluation time: 6.141
```

pas très performant, mais la double boucle est un bon exercice de programmation.

d) Calcul de Π par la formule d' arctan(1)

$$\pi = 4\times\left(\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots+\frac{(-1)^n}{2n+1}+\ldots\right)$$

Cette formule vient de : $Arctan\ x = x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ldots+(-1)^n\ \frac{x^{2n+1}}{2n+1}+\ldots$ pour $|x|\le 1$

et en particulier de Arctan(1) = $\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots+\frac{(-1)^n}{2n+1}+\ldots=\frac{\pi}{4}$

e) Calcul de Π avec la formule de Méchin

$$\pi = 16\ Arctan\frac{1}{5} - 4\ Arctan\frac{1}{239}$$

Exemple 32, algorithme d'Euclide :

Principe pcgd (12,7) = pgcd(5,7)= pgcd(5,2)= pgcd(3,2)=pgcd(1,2) = pgcd(1,0)=1

a et b sont deux entiers
tant que a et b non nuls, si a>=b alors a=a-b sinon b=b-a
si a=0 alors pgcd = b sinon pgcg = a
Rappelons que prouver qu'un algorithme est correct est chose complexe : il ne doit pas planter (opération impossible, accès à un objet inexistant, espace réservé insuffisant – cause de la catastrophe d'Ariane-, …), ne doit pas boucler indéfiniment (démonstration relevant en général de la « descente infinie de Fermat ») et doit fournir le résultat espéré (variante du raisonnement par récurrence).
Ici, cet algorithme ne boucle pas puisque $|a|+|b|$ décroît strictement à chaque passage dans la boucle.

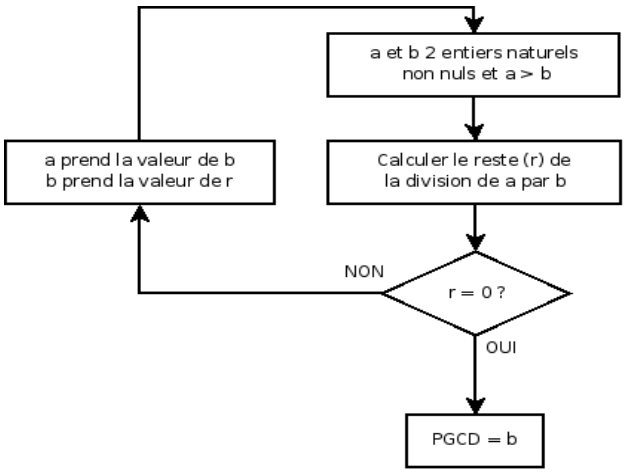
Exemple 33, algorithme récursif d'Euclide :

L'algorithme précédent peut être accéléré

a et b sont deux entiers
tant que a et b non nuls, si a>=b alors a :=a mod b sinon b :=b mod a
si a=0 `alors pgcd := b sinon pgcg := a`

pgcd(10780,3675) = pgcd(3675, 3430) = pgcd(3430,245) = pgcd(245,0) = 245

I



Rappelons que prouver qu'un algorithme est correct est chose complexe : il ne doit pas planter (opération impossible, accès à un objet inexistant, espace réservé insuffisant – cause de la catastrophe d'Ariane-, …), ne doit pas boucler indéfiniment (démonstration relevant en général de la « descente infinie de Fermat ») et doit fournir le résultat espéré (variante du raisonnement par récurrence).
Ici, ces algorithmes ne bouclent pas puisque $|a|+|b|$ décroît strictement à chaque passage dans la boucle.

Une écriture récursive est possible, mais pas avec AlgoBox

Fonction PGCD(a:nombre, b:nombre):nombre

Si b=0 | alors PGCD :=a

Sinon | r égal au reste de la division entière (modulo) de a par b
| PGCD :=PGCD(b, r)

Exemple 34, algorithme de Blankinship pour l'égalité de Bezout :

Exemple 35, algorithmes sur les graphes :

Série ES : Dijkstra, Ford-Bellman, Kruskal …

Exemple 36, algorithmes sur les méthodes de calculs d'intégrales :

Série S : rectangle, trapèze, point moyen, Simpson…

Exemple 37, des algorithmes forts instructifs … :

a) Equations différentielles : méthode d'Euler, accélération de Runge …

b) Equations non linéaires : point fixe, méthode de la sécante, Newton, Delta d'Aitken, Steffesen et Romberg…

c) Systèmes linéaires : Gauss Seidel, Souriau..

d) Algorithme CORDIC.

e) Algorithmes de connexité.

f) La compression d'images

g) Les cryptages

L'analyse numérique est revenue en force avec l'avènement d'ordinateurs rapides. Envoyer un homme dans la Lune est un problème ancien qu'on a mis longtemps de côté tant qu'on n'avait pas inventé de machines capables d'effectuer des millions d'opérations par seconde.

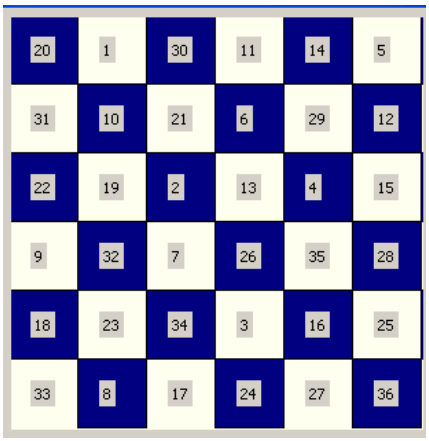
Mais la calcul numérique génère ses propres problèmes : propagation des erreurs, convergence, stabilité des algorithmes… Les habitués du calculs numériques n'écrivent pas les tests entre réels sous la forme « si a ==b » mais « si abs (a-b) < eps » et connaissent les phénomènes d'absorption ($\sum_{i=1}^{n} \frac{1}{i}$ pour des très grandes valeurs de n ne fournit pas le même résultat en allant de 1 vers n, ou de n vers 1).

Exemple 38, des algorithmes aussi pour le plaisir ! :

a) Le compte est bon

b) La promenade du cavalier :

Comment visiter toutes les cases d'un échiquier 8x8 en suivant la règle du cavalier du jeu d'échecs (2 pas en avant, 1 pas de côté) ? Voici une solution Visual Basic sur un damier 6x6 obtenue avec un algorithme de backtracking qui consiste à essayer de progresser en revenant en arrière lors d'une impasse.



c) Les pentaminos :

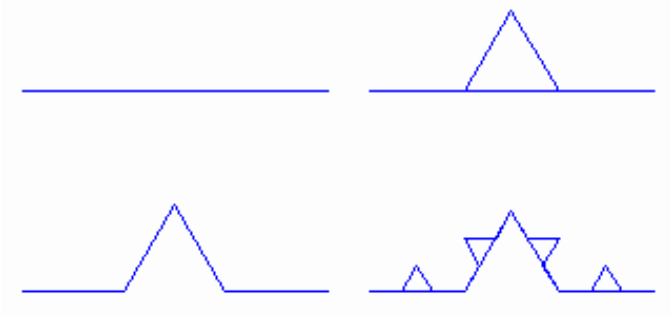
Comment remplir un rectangle 5x12 ou 6x10 avec tous les pentaminos à 5 éléments ?

1	1	1	1	1	12	11	11	11	9	9	9
2	7	7	7	10	12	12	12	11	11	9	4
2	2	2	7	10	10	12	6	6	8	9	4
3	3	2	7	5	10	10	6	8	8	8	4
3	3	3	5	5	5	5	6	6	8	4	4

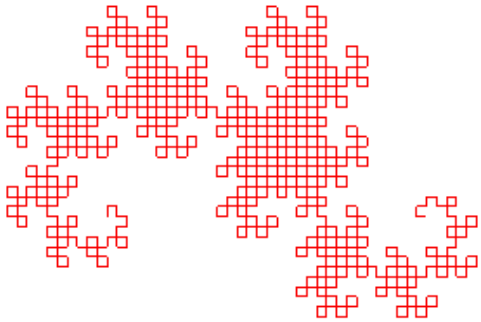
d) Le théorème des 4 couleurs :

Un gros succès récent de la recherche mathématique ; mais comment concrètement colorier une carte donnée ?

e) Les fractales :

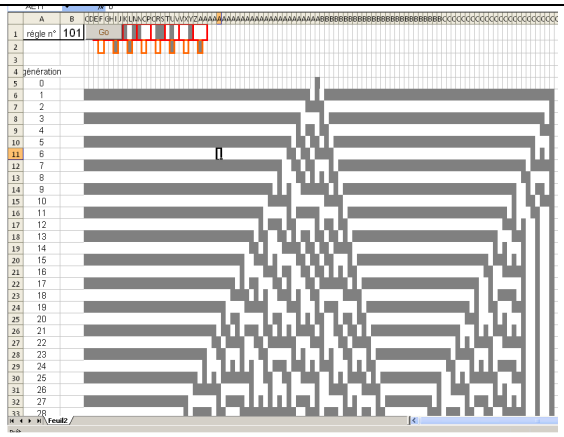


Von Koch : les quatre premières étapes

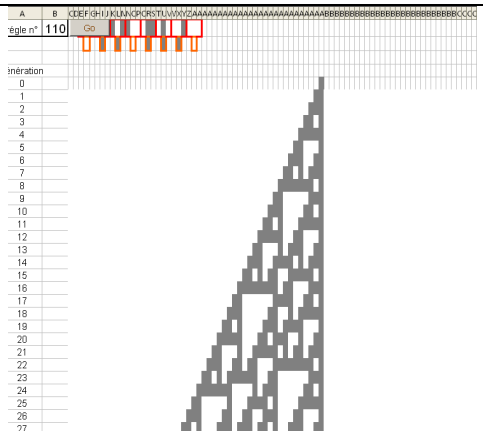


La courbe du dragon

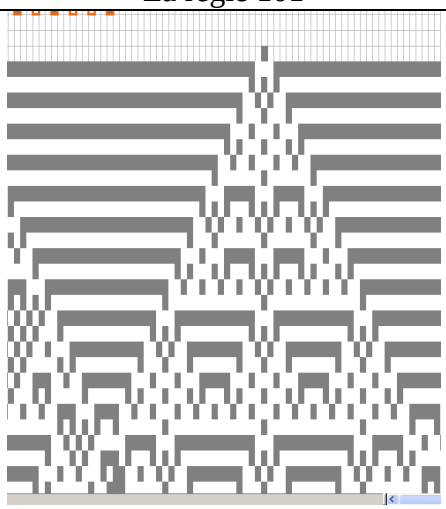
f) Les automates cellulaires ou comment générer du très complexe avec du très simple…
Voici un exemple de domaine des mathématiques où on décrit des phénomènes et on n'a pas de … démonstrations !



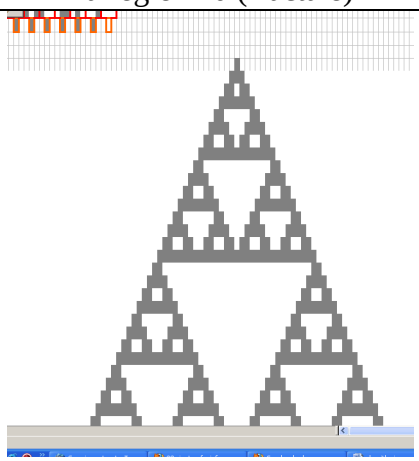
La règle 101



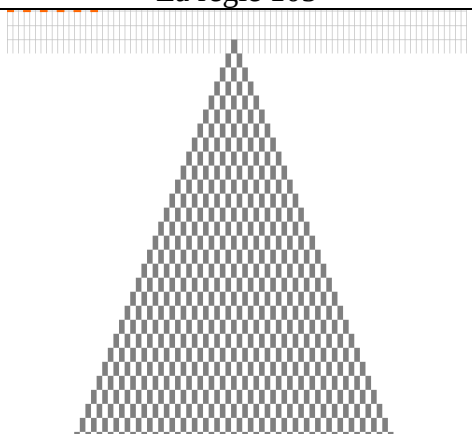
La règle 110 (fractale)



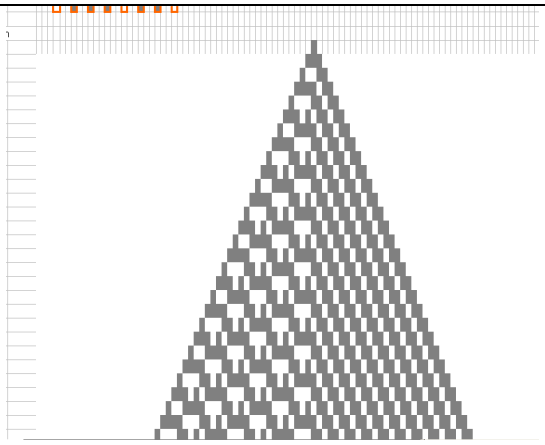
La règle 105



La règle 126 (fractale)



La règle 114 (motif régulier)



La règle 118