Réflexions autour de l'Algorithmique au lycée Mai 2012

Sommaire : **page**

Algorithmes et programmation

► Un algorithme est un processus qui décompose une tâche globale en une succession de tâches élémentaires : ces tâches élémentaires sont en nombre fini et s'effectuent dans un ordre donné.
En bâtissant un algorithme, on a deux préoccupations :

- **l'arrêt de l'algorithme au bout d'un nombre fini d'étapes (convergence)** (Voir exemple en annexe 1)
- **la complexité d'exécution de l'algorithme c'est à dire sa gourmandise en temps et en mémoire.**

► On pourra distinguer trois parties dans un algorithme :

Les entrées (Lire, Saisir)
Le traitement de la tâche décomposée en une succession de tâches élémentaires.
Les sorties (Ecrire, Afficher)

Une déclaration des variables précède en général ces trois parties.

Un algorithme est une succession d'instructions qui permet de résoudre un problème de façon répétitive en un nombre fini d'étapes avec une bonne gestion des entrées et des sorties. On vérifiera qu'il se termine et on tiendra compte de sa complexité (temps d'exécution et mémoire utilisée)

Un algorithme est une suite d'instructions élémentaires qui s'appliquent dans un ordre déterminé à des données et qui fournissent en un nombre fini d'étapes des résultats.

► Concrètement, les algorithmes seront écrits en langage naturel (compréhensible par tous) puis programmés en langage informatique (compréhensible par une machine).
Elaborer un algorithme se fait indépendamment d'une utilisation de la machine. En mathématiques, c'est une stratégie de résolution de problème.
Elaborer un programme c'est traduire l'algorithme en langage informatique. Comme les calculs à la machine peuvent être plus ou moins approximatifs (Voir exemple annexe 2), cela revient à accepter des résultats plus ou moins exacts.
L'objectif de formation visé n'est pas de demander à l'élève de s'adapter à un trop grand nombre de logiciels : il doit déjà se familiariser avec un logiciel de géométrie dynamique, un tableur, un logiciel de calcul formel, une calculatrice et un logiciel d'algorithmique. Parmi les logiciels de programmation, on peut distinguer des logiciels pour l'initiation qui sont en fait des langages intermédiaires de programmation(Algobox) et des outils plus pointus (Xcas, Python,…). Il y a peut-être la nécessité de travailler assez vite avec un logiciel de programmation un peu plus puissant qu'Algobox.

On peut utiliser :

En seconde : Algobox , Xcas,… et éventuellement la calculatrice
Dans les classes supérieures : la calculatrice habituelle complétée par un logiciel de programmation tel que Xcas...

► **Matériel :** on peut travailler
en salle informatique ou avec l'aide d'une classe mobile (chariot d'ordinateurs portables), l'idéal étant d'avoir une salle informatique avec un coin travail sans ordinateur et la possibilité d'accéder aux ordinateurs quand c'est pertinent
ou avec un vidéoprojecteur utilisé par un élève ou par le professeur. L'élève doit aussi avoir la possibilité de travailler en autonomie en étude, au CDI, à l'internat…
Dans tous les cas, il semble inutile de donner à l'élève un parcours trop balisé pour l'utilisation du logiciel : les élèves s'approprient naturellement le logiciel s'il est assez convivial et surtout si le professeur a déjà traité quelques exemples avec l'aide du vidéoprojecteur.

Calculatrice : environnement austère, manipulations longues et fastidieuses, les erreurs de syntaxe sont fatales, gestion des plantages difficile. Beaucoup retrouvent les blocages qu'ils ont en maths. Bien pour ceux qui vont en S.
Avantage : les élèves repartent avec leur programme dans leur cartable.

Algobox : Outil rigoureux, de bonnes capacités, convivial, pas difficile en terme de syntaxe. Pourtant ce langage intermédiaire de programmation risque de ne pas suffire : il faudra évoluer vers un langage plus pointu comme Xcas ou la calculatrice).
Quelques inconvénients signalés par les élèves :

- les « copier coller » sont un peu lourds à mettre en œuvre.
- dans les boucles pour, le pas n'est pas réglable.
- l'instruction pow(x,n) n'est pas conviviale

Xcas : environnement austère mais performant et rigoureux. C'est un logiciel qui présente une grande fiabilité en calcul numérique et qui comporte le calcul formel intégré. L'instruction d'affectation est :=

Scratch : performant mais relativement lent, attrayant, allure peu professionnelle, un peu flashi, très visuel en géométrie. Le professeur doit être vigilant car beaucoup d'élèves sont tentés de passer leur temps à changer le lutin ou à utiliser le son.

Python, Scilab: pas encore vraiment testés.

Pourquoi l'algorithmique dans les programmes ?

► L'étude de l'algorithmique sert à former la rigueur et la concentration.
L'algorithmique peut intervenir dans tous les champs du programme.

▪ L'algorithmique développe des capacités suivantes : logique, rigueur, modélisation, analyse de problème, enchaînement de raisonnements et de calculs, organisation des blocs de répétition, construction de structures, compréhension des boîtes noires…

▪ La programmation développe d'autres capacités : rigueur de l'écriture, rigueur de la structure, nécessité d'obtenir un résultat, pratique des tests, de la vérification et du contrôle (auto-contrôle)...

► Utiliser de façon pertinente un logiciel pertinent.
Un logiciel de programmation d'algorithmes n'est ni un traceur de courbe, ni un logiciel de géométrie dynamique ni un logiciel de calcul formel. La programmation d'algorithmes sur une machine peut créer

des problèmes dans le traitement des nombres (certains calculs ainsi que le codage en binaire des nombres réels induisent une imprécisions sur les résultats donnés par la machine). Dans le cadre de l'enseignement, il est important d'identifier les cas où la mobilisation d'un logiciel de programmation d'algorithmique est pertinente. En seconde, apprendre aux élèves à penser algorithmiquement un problème semble pertinent dans les cas où on doit mettre en place un calcul automatisé, mobiliser des fonctions à deux variables, des fonctions affines par morceaux, faire des simulations en probabilités, comprendre le fonctionnement d'une boite noire (Voir exemple en annexe 4: fonction partie entière), Traiter des situations définies par récurrence.
On trouvera en activité 25 un **exemple où la mobilisation de l'algorithmique est incontournable**.
Il est important de mettre en évidence les limites de l'outil utilisé. (Voir exemple en annexe 2 de « point sur cercle »)

► L'algorithmique peut permettre de faire le tour d'une notion (en conduisant de façon naturelle les élèves à aborder les cas particuliers), de faire un bilan en prenant de la hauteur : **c'est un outil de formalisation et de synthèse a posteriori qui s'intègre fort bien à l'accompagnement personnalisé.**
Exemple en seconde : Ecrire un algorithme qui fournit une équation de la droite (AB) quand on saisit en entrée, les coordonnées des points A et B.
Exemple en première : Ecrire un algorithme qui donne en sortie le nombre de solutions strictement positives d'une équation du second degré du type $ax^2+bx+c=0$ quand on saisit, en entrée, les réels a, b, c.
Exemple en première : Ecrire un algorithme qui permet de tester la position relative deux droites (parallèles ou non, perpendiculaires ou non) quand on saisit, en entrée, les réels a, b , c, a',b',c' correspondants aux équations cartésiennes $ax+by+c=0$ et $a'x+b'y+c'=0$ des deux droites.

►L'algorithmique permet souvent **de faire vivre et d'accompagner des problèmes ouverts**, **d'aborder des problèmes nouveaux ou de les poser autrement.**

►L'algorithmique est véritablement une partie des mathématiques qui produit elle-même des problèmes.

Structures élémentaires d'un algorithme

Au lycée, on se limite à la programmation impérative : on ne traite pas la programmation fonctionnelle ni la programmation récursive donc pas la récursivité.
Exemple de récursivité le quotient entier avec Xcas: qe(a,b) := si a<b alors 0 ; sinon 1+qe(a-b,b) ; fsi ;
On évite au maximum les Lbl GoTo pour **se concentrer sur les structures modulaires suivantes :**

► **Succession de calculs :** on trouvera dans ces algorithmes des instructions d'**affectation**
(la variable X prend pour valeur *a* ou *X :=a* ou *X←a* ou Affecter la valeur *a* à la variable *X* ou *a→X*)

► **Structure conditionnelle :**
- **SI, Alors, Fsi (test simple)**
  Si *proposition booléenne*, alors *action*, FinSi
- **SI, Alors, Sinon, FinSi (test complet)**
  Si *proposition booléenne*, alors *action1 (cas vrai)*, sinon *action2 (cas faux)*, FinSi

► **Structures itératives : boucles ou répétition d'un groupe d'instructions**
- **Pour :** Bien adapté quand le nombre d'itérations est connu
  Pour *compteur* allant de *début* à *fin*, faire *action*, FinPour

- **Tant que :** Bien adapté quand le nombre d'itérations est inconnu.
  Tant que *proposition booléenne* vraie, Faire action, FinTantQue

  Attention : il est indispensable que *proposition booléenne* finisse par devenir fausse pour que l'algorithme s'arrête. (Voir, en annexe 1, le tirage aléatoire de deux noms d'élèves distincts).

Remarque : une boucle « Tant que » pour laquelle on connaît le nombre d'itérations peut être remplacée par une boucle « Pour » en introduisant un compteur. Mais la réciproque est fausse.

▪ **Répéter jusqu'à *proposition booléenne* vraie**
Contrairement au « tant que », la *proposition booléenne* est placée à la fin.
Cette instruction, qui n'est pas disponible sur tous les logiciels, n'est pas une priorité de la classe de seconde.

Progression dans l'année

► Commencer l'algorithmique tôt dans l'année et avancer progressivement.
Exiger une écriture en langage naturel puis une écriture en langage intermédiaire de programmation.
Choisir une présentation problématisée pour les différentes structures (une structure est introduite à partir d'une situation problème quand le besoin s'en fait sentir).
A partir d'une écriture en langage naturel, on peut envoyer des élèves au tableau « jouer » le déroulement de l'algorithme c'est à dire qui vont faire tourner l'algorithme à la main.

► Un exemple de progression
Pour introduire, comprendre et mettre en œuvre une nouvelle instruction d'algorithmique, il me semble préférable d'utiliser des activités qui prennent appui sur des supports non mathématiques ou sur des supports mathématiques bien naturalisés par les élèves.

Exemple de progression en seconde 2010-2011 :
- Insister sur la gestion des entrées et des sorties (les identifier et les gérer).
- Introduire l'affectation X prend pour valeur *a* (en abrégeant : X ppv *a*).Distinguer affectation et égalité.
- Instruction conditionnelle (non imbriquées)
  - 2 éventualités avec Si Alors puis à nouveau SI Alors
  - 2 éventualités avec Si Alors SINON
- Instruction TANT QUE pour contrôler une saisie.
- ET / OU
- Instructions conditionnelles imbriquées.
- Répétitions quand le nombre d'itérations est connu.
- Introduction d'un compteur (K prend pour valeur K+1)
- Auto affectation : K prend pour valeur K+e

L'instruction « répéter jusqu'à », les chaînes de caractères, les listes ne sont pas des attendus du programme.

Traces écrites

On peut distinguer l'algorithmique en tant qu'objet et l'algorithmique en tant qu'outil.

Algorithmique en tant qu'outil : ici l'algorithmique sert à résoudre des problèmes : il n'y a pas lieu d'isoler ces traces écrites. C'est-à-dire lorsque l'outil algorithmique intervient de façon pertinente dans la résolution d'un problème, la trace écrite reste associée au problème et ne doit pas être isolée. Il ne semble pas utile de donner un document trop précis concernant le mode d'emploi du logiciel : l'élève a vite fait de s'approprier le logiciel surtout s'il a vu le professeur l'utiliser en classe au vidéoprojecteur par exemple.

Algorithmique en tant qu'objet : entrées et sorties, affectations, structures…On peut faire des synthèses dans un chapitre du cours qui s'ouvre en début d'année et qui se complète au fur et à mesure.

Evaluation

► Rappel du document de Françoise Munck :

« Il s'agit essentiellement d'une évaluation par compétences en valorisant :
le choix de mobiliser la démarche algorithmique quand elle est pertinente,
les compétences liées davantage à la démarche qu'à la seule maîtrise des logiciels,
les compétences d'auto-contrôle (identification et rectification d'une erreur)
Il semble indispensable que le professeur puisse intervenir pendant l'évaluation par exemple pour valider un résultat ou faire identifier un problème et permettre à l'élève d'aller plus loin, pour l'engager à regarder plus attentivement tel ou tel point, pour donner une indication d'ordre logiciel etc …

Deux types d'évaluation possibles :
• évaluation spécifique avec ou sans machine (style épreuve de bac L) : interprétation ou rectification d'un algorithme déjà écrit, écriture d'un algorithme répondant à une commande spécifique. Cela entre parfaitement dans le cadre d'une évaluation « ordinaire » de mathématiques. Dans le cadre d'une épreuve classique avec calculatrice, le professeur n'intervient pas.
• évaluation de type épreuve pratique, avec un problème complexe à résoudre pour lequel la programmation d'un algorithme est une démarche possible, à l'initiative de l'élève. Cette utilisation ou non de l'algorithmique peut dépendre de la modélisation choisie.

Par ailleurs, au niveau de l'évaluation formative, il ne faut pas négliger l'introduction de parties algorithmique dans les devoirs à la maison (logiciels à la maison, au lycée, calculatrices) ».

► **Compétences attendues des élèves en algorithmique:**
A travers la résolution de problèmes :
- Faire fonctionner un algorithme et identifier ce qu'il produit.
- Interpréter un algorithme et le modifier.
- Etre capable de choisir la démarche algorithmique et d'écrire un algorithme simple pour résoudre un problème donné puis programmer cet algorithme.
- Modifier la programmation d'un algorithme pour le corriger ou pour l'adapter afin de résoudre un problème voisin.
- Pratiquer systématiquement la vérification et le contrôle.

Tester un algorithme (essais « au hasard », puis essais organisés en soulignant l'importance d'une analyse logique et rigoureuse). Savoir détecter un algorithme erroné ou un algorithme qui ne se termine pas. Tester un algorithme peut permettre de montrer qu'il contient des erreurs mais le fait d'obtenir de bons tests ne prouve pas que l'algorithme soit juste. Prouver qu'un algorithme est juste revient à l'analyser avec logique et rigueur : il doit être juste par construction. Voir en annexe 3 un algorithme de tri de trois notes. On obtient un algorithme faux et on découvre la « démonstration » d'un algorithme.

► Pour évaluer la maitrise des attendus en algorithmique, on peut envisager un devoir surveillé en salle informatique surtout si on veut évaluer l'aptitude à programmer un algorithme: il y a souvent nécessité de prévoir plusieurs sujets pour éviter les tricheries. Le professeur n'intervient pas directement mais il donne bien sûr « gratuitement » les indications techniques concernant le logiciel. Il peut être intéressant de prévoir des aides pénalisantes ou non : ces aides peuvent être orales donc très personnalisées (l'élève rencontre le professeur, lui expose son problème et attend une aide en abandonnant, par exemple, un demi-point) ou elles peuvent être écrites, préparées à l'avance, question par question, et disposées au fond de la classe (l'élève va chercher une aide pour une question donnée et abandonne par exemple un demi-point).

► On peut aussi conduire une évaluation à travers une épreuve pratique pour laquelle on abandonnera la notation traditionnelle qui n'a pas de sens si on veut s'attacher aux démarches, aux tentatives de résolution, aux stratégies de contrôle… Même si les élèves et les familles restent attachés à un résultat chiffré (dans ce cas, on peut donner un bonus sur la moyenne : pas très original mais efficace et juste), il faut passer à une évaluation de compétences.

Dans ce cas, le professeur intervient dans la séance (il valide certains résultats, il apporte une aide technique, il questionne, il encourage et motive l'élève. Le professeur utilise une grille d'évaluation qu'il doit remplir en tournant dans la classe (ce qui n'est pas facile avec une demi-classe) avec des objectifs qui ont été auparavant donnés et expliqués aux élèves. Il peut évaluer les points suivants :

Motivation, initiatives devant une situation de recherche.
Analyse du sujet
Mobilisation des connaissances
Mise en place des stratégies
Questionnements
Démarches de contrôle et de vérification.
Compte-rendu ou qualité dans la restitution des résultats.

Il veillera à laisser assez de temps aux élèves et à différencier les activités.

► Ce point de vue s'inscrit dans une diversification de l'évaluation en seconde :

▪ Devoirs surveillés classiques, restitution d'acquis, QCM, questions avec prise d'initiatives (avec valorisation des écrits intermédiaires), évaluation à l'oral en lien avec un travail écrit…

▪ Devoirs maison avec possibilité d'utiliser les ordinateurs au CDI, en étude, à l'internat…

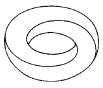
▪ Evaluation de travaux pratiques utilisant ou non les TICE avec remise d'un compte rendu écrit, d'une narration de recherche, d'un fichier, avec ou sans aide orale du professeur, avec ou sans aide écrite de la part du professeur.

Le professeur est toujours disponible pour aider à la maîtrise technique des logiciels.

▪ Evaluation par compétences qui peut ne pas conduire à une note chiffrée. Le professeur peut intervenir durant cette évaluation et il peut s'l le souhaite s'aider d'une grille.

► **Exemples Compétences et évaluation ac-nancy-metz**

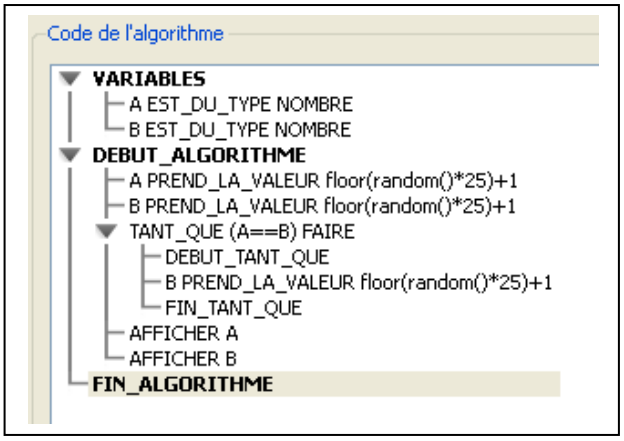
Compétences évaluées	Critères et indicateurs	Niveaux de maîtrise et degré d'autonomie
Analyser la situation étudiée (langage naturel)	**Examen et traduction du problème** ▪ Proposition d'une démarche de résolution ▪ Ebauche d'une démarche algorithmique en langage naturel	▪ L'élève propose seul une démarche cohérente de résolution ▪ L'élève fait preuve d'initiative et propose une démarche de résolution ▪ L'élève traduit le problème à l'aide d'un indice donné par le professeur
Mettre au point une solution algorithmique	**Détermination de la structure de l'algorithme** ▪ Identification des données d'entrée ▪ Identification des résultats de sortie ▪ Identification des variables ▪ Détermination d'une séquence logique des "opérations" ▪ Détermination des types de traitement appropriés à chacune des opérations (séquentiel, conditionnel, itératif)	▪ L'élève trouve seul les entrées, sorties et variables pertinentes ▪ L'élève détermine seul la structure du traitement
Déterminer la validité d'un algorithme Comprendre et analyser un algorithme éventuellement fourni Valider une solution algorithmique	**Vérification de la pertinence de la solution** ▪ Traces d'exécution ▪ Détermination des erreurs et des lacunes de la solution algorithmique mise au point ▪ Vérification de la terminaison d'un traitement **Modification appropriée de la solution algorithmique** ▪ Correction des erreurs ▪ Proposition d'une amélioration : généralisation, meilleure approximation...	▪ L'élève fait preuve d'initiative dans le choix des éléments de contrôle ▪ L'élève corrige les erreurs en suivant les indications du professeur ▪ L'élève sait repérer seul les erreurs et les corriger ▪ L'élève fait preuve d'initiative pour les extensions demandées.
Présenter sa démarche, les résultats obtenus	**Mise en forme de la production** ▪ Commodité et clarté de l'interface utilisateur (entrées - sorties) ▪ Compte-rendu soigné et lisible du travail demandé ▪ Présence de toute l'information nécessaire à l'interprétation de l'algorithme	▪ La production contient des demandes d'entrées pertinentes, des lignes de commentaires judicieuses pour la compréhension du fonctionnement du programme ▪ La production contient des défauts de lisibilité dans l'interface ou dans les commentaires.



Annexes

Annexe 1

Analyser l'algorithme suivant.



▫ *Cet algorithme (ou plutôt ce programme puisqu'il est écrit sous Algobox) répond au problème suivant :*
« Tirer au hasard les noms de deux élèves distincts pour qu'ils aillent en même temps au tableau »
L'intérêt de cet exemple est de discuter sur la condition d'arrêt de cet algorithme.
En pratique, la condition B = = A finira bien par se réaliser mais en théorie il est possible que cet algorithme ne se finisse jamais.
On pourrait dire ici que l'algorithme n'est pas correct alors que le programme fonctionne...

▫ *Si des collègues souhaitent avoir un algorithme correct qui réponde au problème, on pourra donner le suivant (court mais difficile à expliquer en seconde).*

A prend pour valeur Alea(1,25)
B prend pour valeur Alea(1,24)
Si B est supérieur ou égal à A alors B prend la valeur B+1
Afficher A et B.

▫ *On retrouve cette difficulté dans les tirages de boules dans une urne sans remise (voir activité 20 en 1S)*
▫ *On retrouve cette problématique dans le bac S Pondichéry avril 2012.*

Annexe 2

On se donne un point *A* par ses coordonnées, un nombre positif R et un point *M* également par ses coordonnées.
On se demande si le point *M* appartient au cercle de centre *A* et de rayon R.

Avec A(0 ;0) R=2 *et M*(sqrt(3) ;1), *le programme* 1 *ne marche pas alors que le programme* 2 *fonctionne.*

Programme 1 ALGOBOX

```
1     VARIABLES
2       xa EST_DU_TYPE NOMBRE
3       ya EST_DU_TYPE NOMBRE
4       R EST_DU_TYPE NOMBRE
5       xm EST_DU_TYPE NOMBRE
6       ym EST_DU_TYPE NOMBRE
7       T EST_DU_TYPE NOMBRE
8     DEBUT_ALGORITHME
9       AFFICHER "Abscisse de A"
10      LIRE xa
11      AFFICHER "Ordonnée de A"
12      LIRE ya
13      AFFICHER "Rayon du cercle"
14      LIRE R
15      AFFICHER "Abscisse du point M"
16      LIRE xm
17      AFFICHER "Ordonnée de M"
18      LIRE ym
19      T PREND_LA_VALEUR pow((xm-xa),2)+pow((ym-ya),2)-R*R
20      SI (T==0) ALORS
21        DEBUT_SI
22        AFFICHER "M est sur le cercle"
23      FIN_SI
24      SINON
25          DEBUT_SINON
26          AFFICHER "M n'est pas sur le cercle"
27      FIN_SINON
28    FIN_ALGORITHME
```

Programme 2 ALGOBOX

```
1     VARIABLES
2       xa EST_DU_TYPE NOMBRE
3       ya EST_DU_TYPE NOMBRE
4       R EST_DU_TYPE NOMBRE
5       xm EST_DU_TYPE NOMBRE
6       ym EST_DU_TYPE NOMBRE
7       T EST_DU_TYPE NOMBRE
8     DEBUT_ALGORITHME
9       AFFICHER "Abscisse de A"
10      LIRE xa
11      AFFICHER "Ordonnée de A"
12      LIRE ya
13      AFFICHER "Rayon du cercle"
14      LIRE R
15      AFFICHER "Abscisse du point M"
16      LIRE xm
17      AFFICHER "Ordonnée de M"
18      LIRE ym
19      T PREND_LA_VALEUR pow((xm-xa),2)+pow((ym-ya),2)
20      SI (T==R*R) ALORS
21        DEBUT_SI
22        AFFICHER "M est sur le cercle"
23      FIN_SI
24      SINON
25          DEBUT_SINON
26          AFFICHER "M n'est pas sur le cercle"
27      FIN_SINON
28    FIN_ALGORITHME
```

Cet exemple (qui n'est pas très intéressant mathématiquement) montre deux algorithmes qui, d'un strict point de vue algorithmique, sont valables et équivalents. Quand on passe à la programmation, il y a des cas où l'un des deux algorithmes donne le bon résultat et pas l'autre. Il faudrait expliquer aux élèves qu'en informatique il y a des problèmes d'arrondis et pour avoir un algorithme fonctionnant dans tous les cas (ou presque, il faudrait remplacer dans le premier algorithme la condition T = = 0 par abs(T)<10^{-6} (ou par : T<10^{-6} ET T>–10^{-6})
Ce problème se rencontrera dès qu'il y a des problèmes d'arrondis dans une condition : c'est la grosse différence entre algorithmique théorique et programmation.

Voici la programmation de cet algorithme sous XCas. On remarque que la syntaxe n'est pas plus compliquée et que cet outil, plus professionnel, doté d'un meilleur logiciel de calcul et doublé d'un logiciel de calcul formel donne de meilleurs résultats.

```
Programme 3 Xcas
saisir("xa",xa);
saisir("ya",ya);
saisir("rayon",R);
saisir("xm",xm);
saisir("ym",ym);
D:=(xm-xa)^2+(ym-ya)^2-R^2;
si D==0 alors afficher("oui");
sinon afficher("non");
fsi;
```

Annexe 3

Que fait cet algorithme ?

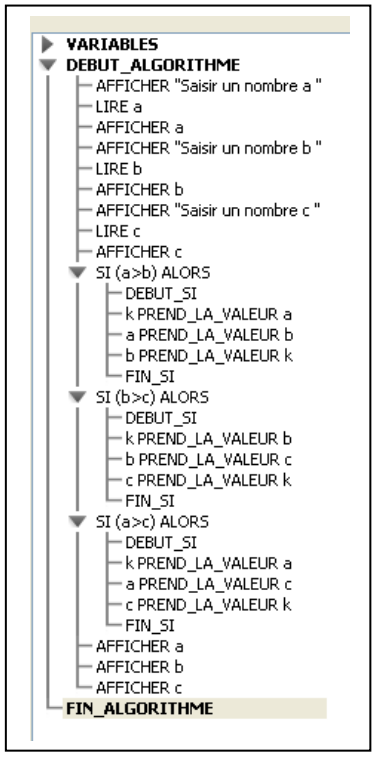
Lire a,b,c (nombres distincts)

Si a>b alors
Echanger a et b

Si b>c alors
Echanger b et c

Si a>c alors
Echanger a et c

Ecrire a,b,c



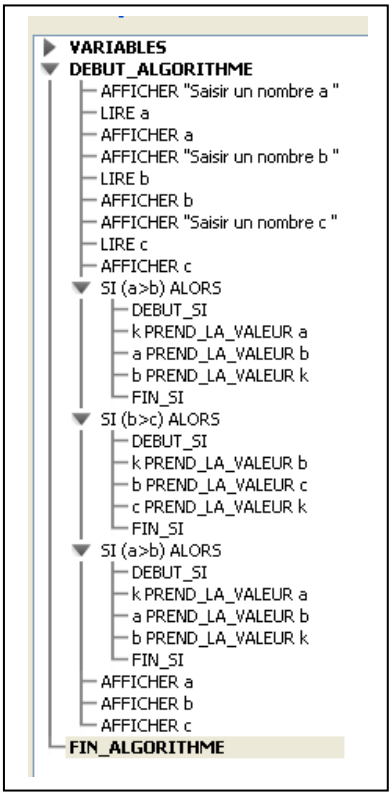
L'intérêt de cette recherche est de montrer comment tester un algorithme et aussi comment « démontrer » qu'un algorithme effectue bien la tâche demandée.

■ *On peut tester cet algorithme en faisant des essais au hasard.*
ex 1,7,5 5,3,8 8,1,4 etc…. et se convaincre que cet algorithme classe trois nombres par ordre croissant
■ *On peut tester cet algorithme en faisant des essais organisés :*
ex 1,5,7 1,7,5 5,1,7 5,7,1 7,1,5 7,5,1
On constate alors que cet algorithme ne classe pas toujours les trois nombres par ordre croissant (il ne remplit pas cette tâche quand « c» est le plus petit des trois nombres).

■ *Comment modifier cet algorithme pour qu'il classe les trois nombres par ordre croissant ?*

Voici une idée

```
Entrée : a,b,c : Entier
Sortie : a,b,c : Entier

Début
si a>b alors

echanger(a,b)
finsi

si b>c alors

echanger(b,c)
finsi

si a>b alors

echanger(a,b)
finsi

fin
```



■ *On peut se demander comment être sûr que cet algorithme effectue bien la tâche demandée. Pour répondre à cette question , on peut proposer une preuve en examinant tous les cas .*
En appelant n_1 , n_2 , n_3 *les trois nombres saisis dans cet ordre,*
on examinera les six cas :
$n_1 < n_2 < n_3$ $n_1 < n_3 < n_2$ $n_2 < n_1 < n_3$ $n_2 < n_3 < n_1$ $n_3 < n_1 < n_2$ $n_3 < n_2 < n_1$

Annexe 4 Peut-on faire confiance à Algobox ?

Enoncé :
Ecrire un algorithme qui demande de saisir trois points *A*, *B*, *C* par leurs coordonnées et qui indique si les trois points saisis sont alignés. Programmer cet algorithme sous Algobox et l'envoyer à : gerard.cordes@ac-nantes.fr

Dans l'algorithme suivant, Paul a utilisé le critère de colinéarité de 2 vecteurs. Suivant l'ordre de saisie des trois points, on a deux résultats contradictoires.

ALGOBOX : PAUL B.

```
1     VARIABLES
2       a EST_DU_TYPE NOMBRE
3       b EST DU TYPE NOMBRE
4       c EST DU TYPE NOMBRE
5       xa EST DU TYPE NOMBRE
6       ya EST_DU_TYPE NOMBRE
7       xb EST_DU_TYPE NOMBRE
8       yb EST_DU_TYPE NOMBRE
9       xc EST DU TYPE NOMBRE
10      yc EST DU TYPE NOMBRE
11      AB EST_DU_TYPE NOMBRE
12      AC EST_DU_TYPE NOMBRE
13    DEBUT ALGORITHME
14      AFFICHER "A : ("
15      LIRE xa
16      AFFICHER xa
17      AFFICHER ";"
18      LIRE ya
19      AFFICHER ya
20      AFFICHER ")"
21      AFFICHER "B : ("
22      LIRE xb
```

```
23      AFFICHER xb
24      AFFICHER ";"
25      LIRE yb
26      AFFICHER yb
27      AFFICHER ")"
28      AFFICHER "C : ("
29      LIRE xc
30      AFFICHER xc
31      AFFICHER ";"
32      LIRE yc
33      AFFICHER yc
34      AFFICHER ")"
35      SI ((xb-xa)*(yc-ya)==(yb-ya)*(xc-xa)) ALORS
36        DEBUT_SI
37        AFFICHER "A,B et C sont alignés."
38        FIN_SI
39        SINON
40          DEBUT SINON
41          AFFICHER "A,B et C ne sont pas alignés."
42          FIN_SINON
43    FIN_ALGORITHME
```

RESULTATS :

```
***Algorithme lancé***
A : (8;-0.6)
B : (0;1)
C : (5;0)
A,B et C ne sont pas alignés.
***Algorithme terminé***
***Algorithme lancé***
A : (5;0)
B : (0;1)
C : (8;-0.6)
A,B et C sont alignés.
***Algorithme terminé***
```

Dans les deux algorithmes suivants, les élèves ont choisi de travailler avec les coefficients directeurs d'une droite.
Van-Tin n'a pas été assez rigoureux : il est sanctionné car l'algorithme refuse la division par zéro quand on saisit : A(2,2) B(2.3) C(2.3).
Marine n'a pas été plus rigoureuse mais son algorithme, rédigé différemment, n'est pas interrompu : il accepte de tester l'égalité de deux quantités résultant d'une division par zéro.

ALGOBOX : VAN-TIN

```
1     VARIABLES
2       xA EST_DU_TYPE NOMBRE
3       yA EST_DU_TYPE NOMBRE
4       xB EST_DU_TYPE NOMBRE
5       yB EST DU TYPE NOMBRE
6       xC EST DU TYPE NOMBRE
7       yC EST_DU_TYPE NOMBRE
8       m1 EST_DU_TYPE NOMBRE
9       m2 EST_DU_TYPE NOMBRE
10    DEBUT ALGORITHME
11      LIRE xA
12      LIRE yA
13      LIRE xB
14      LIRE yB
15      LIRE xC
16      LIRE yC
17      m1 PREND_LA_VALEUR (yB-yA)/(xB-xA)
18      m2 PREND LA VALEUR (yC-yA)/(xC-xA)
19      SI ((m1==m2)) ALORS
20        DEBUT_SI
21        AFFICHER "Les 3 points A , B et C sont alignés"
22        FIN SI
23        SINON
24          DEBUT_SINON
25          AFFICHER "Les 3 points A , B et C ne sont pas alignés"
26          FIN_SINON
27    FIN_ALGORITHME
```

RESULTATS :

```
***Algorithme lancé***
***Algorithme interrompu ligne 17 : erreur de calcul***
```

ALGOBOX : MARINE

CODE DE L'ALGORITHME :

```
1     VARIABLES
2       xa EST_DU_TYPE NOMBRE
3       ya EST_DU_TYPE NOMBRE
4       xb EST_DU_TYPE NOMBRE
5       yb EST DU TYPE NOMBRE
6       xc EST DU TYPE NOMBRE
7       yc EST DU TYPE NOMBRE
8     DEBUT_ALGORITHME
9       LIRE xa
10      LIRE ya
11      LIRE xb
12      LIRE yb
13      LIRE xc
14      LIRE yc
15      SI ((yb-ya)/(xb-xa)==(yc-ya)/(xc-xa)) ALORS
16        DEBUT SI
17        AFFICHER "Les points A B C sont alignés "
18        FIN_SI
19        SINON
20          DEBUT_SINON
21          AFFICHER "Les points ne sont pas alignés "
22          FIN SINON
23    FIN_ALGORITHME
```

RESULTATS :

```
***Algorithme lancé***
Les points A B C sont alignés
***Algorithme terminé**
```

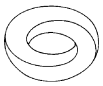



Tableau comparatif des différentes activités expérimentées en classe 2010/2012

Activité	Classe	Fonctions	Fonctions de plusieurs variables	Simulations Probabilités	Géométrie repérée	Suites	Boite noire	Passage à l'algorithme incontournable	Comprendre un algorithme	Modifier un algorithme	Créer un algorithme	Problème ouvert	Evaluation	AP	Titre
1	2 ou 1			×							×	×			Le six en trois coups
2	2 ou 1			×	×						×				Atteindre la cible
3	2	×							×	×			×		Fred prend le taxi
4	2										×	×			Distributeur de billet
5	2		×								×		×	×	Distance d'arrêt
6	2		×								×				Indice de Masse Corporelle
7	2										×	×			Feux tricolores
8	2				×	×					×	×			Fils, carrés, escalier, spirale
9	2 ou 1	×							×	×				×	Enchaînement
10	2 ou 1			×							×	×			Bon générateur?
11	2 ou 1			×					×	×					Déplacement d'un pion
12	2 ou 1 ou T			×			×		×	×					Algorithme de Sébastien
13	2			×					×	×	×		×		Boules et dé
14	2								×	×	×		×		Trois algorithmes en 2nde
15	1 ou T					×					×	×			Nénuphars
16	2								×					×	Fonction définie par un algo
17	2 ou 1														Le plus grand nombre
18	1					×		×			×				Une suite étrange
19	2		×								×				Le test de Ruffier
20	1			×										×	Les bonbons de Paul
21	2 ou 1	×					×		×	×					La boîte noire PE
22	1	×			×					×				×	Second degré
23	T								×				×		La boite noire !
24	1			×							×	×			Une souris dans les tuyaux
25	2 ou 1				×			×							Comptons les points

Activité 1 (le SIX en trois coups)

Voici un jeu : on lance un dé cubique bien équilibré jusqu'à ce que le six sorte.
Si le six est sorti avant le troisième coup ou au troisième coup, c'est gagné sinon c'est perdu.

Questions :

Ecrire un algorithme pour simuler 1000 fois ce jeu.

Estimer la probabilité de gagner à ce jeu .

Activité testée en seconde. Eléments de réponse

Sortir 6 en 3 coups (simulation de N parties)

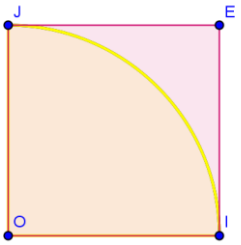
Code de l'algorithme

```
VARIABLES
    D EST_DU_TYPE NOMBRE
    K EST_DU_TYPE NOMBRE
    S EST_DU_TYPE NOMBRE
    N EST_DU_TYPE NOMBRE
    G EST_DU_TYPE NOMBRE
    P EST_DU_TYPE NOMBRE
    L EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE N
    G PREND_LA_VALEUR 0
    POUR L ALLANT_DE 1 A N
        DEBUT_POUR
        S PREND_LA_VALEUR 0
        POUR K ALLANT_DE 1 A 6
            DEBUT_POUR
            D PREND_LA_VALEUR floor(6*random()+1)
            SI (D==6) ALORS
                DEBUT_SI
                S PREND_LA_VALEUR S+1
                FIN_SI
            FIN_POUR
        SI (S!=0) ALORS
            DEBUT_SI
            G PREND_LA_VALEUR G+1
            FIN_SI
        FIN_POUR
    P PREND_LA_VALEUR G/N
    AFFICHER P
FIN_ALGORITHME
```

$1-(5/6)^3 \approx 0.42$

$1-(5/6)^6 \approx 0.66$

Activité 2 (Atteindre une cible)



Sur une planche carrée de côté 1m, on colorie en jaune le quart de disque comme sur la figure ci-contre.
Cette planche devient la cible sur laquelle Antonin envoie des fléchettes au hasard mais sans jamais rater la planche. On souhaite estimer la probabilité pour qu'Antonin atteigne la zone coloriée en jaune.

1. On se place dans le repère orthonormé (O ; I ; J) donné sur la figure.
On choisit de simuler un lancer de fléchette par le choix au hasard de deux réels dans [0 ; 1].
Le point d'impact de la fléchette sera repéré par les coordonnées (x ;y).
a)Donner un critère sur x et y permettant de savoir si le quart de disque colorié en jaune a été atteint ou non.
b) Simuler plusieurs lancers et estimer la probabilité pour qu'Antonin atteigne la zone coloriée en jaune.

2. En écrivant la probabilité cherchée sous la forme du quotient de deux aires, calculer la valeur exacte pour qu'Antonin atteigne le quart de disque colorié en jaune.

Activité testée en seconde

Activité 3 (Fred prend le taxi)

Pour une course en taxi, Fred hésite entre deux tarifs:
Tarif 1 : prise en charge de 2.40€ au départ puis 1.20€ par kilomètre.
Tarif 2 : prise en charge de 4.5€ au départ puis 0.90€ par kilomètre.
On appelle *x* le nombre de kilomètres parcourus en taxi.

1.Exprimer, en fonction de *x*, le montant $f(x)$ de la facture quand on utilise le Tarif 1.

2.Exprimer, en fonction de *x*, le montant $g(x)$ de la facture quand on utilise le Tarif 2.

3.Voici un algorithme (programmé sous Algobox) qui peut aider Fred à choisir entre les deux tarifs :

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    T1 EST_DU_TYPE NOMBRE
    T2 EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    AFFICHER "Saisir le nombre de km     "
    LIRE x
    AFFICHER x
    T1 PREND_LA_VALEUR .....
    T2 PREND_LA_VALEUR .....
    AFFICHER "Montant en euros de la course avec le tarif 1     "
    AFFICHER T1
    AFFICHER "Montant en euros de la course avec le tarif 2     "
    AFFICHER T2
    SI (T1<T2) ALORS
        DEBUT_SI
        AFFICHER "Le tarif .... est plus avantageux"
        FIN_SI
        SINON
            DEBUT_SINON
            AFFICHER "Le tarif ..... est plus avantageux"
            FIN_SINON
FIN_ALGORITHME
```

a) Compléter les quatre parties manquantes dans l'algorithme précédent.
b) Qu'affiche l'algorithme si *x* = 6 ?
c) Qu'affiche l'algorithme si *x* = 10 ?
4.a)Existe-t-il une valeur de *x* pour laquelle les deux tarifs donnent la même facture ?
b) Modifier l'algorithme précédent pour qu'il tienne compte du résultat trouvé au 4a).

Evaluation en 2Bds13 (devoir surveillé : pas d'accès aux ordinateurs)

Activité 4 (Distributeur de billets)

Un distributeur de billets de banque doit donner la somme S demandée
avec des billets de 10, 20 ou 50 € et avec le moins de billets possibles.
La somme demandée doit être multiple de 10 et inférieure ou égale à 500.

On demande à l'utilisateur la somme qu'il souhaite retirer et on voudrait faire afficher sur l'écran du distributeur le nombre de billets de 10, 20 ou 50 qui seront distribués par la machine. Comment faire ?

Activité testée en seconde

Activité 5 (Distance d'arrêt)

La distance d'arrêt D_A en mètres d'un véhicule roulant à la vitesse V (en km/h) peut se calculer par la formule $D_A = \frac{V}{3.6} + \frac{V \times V}{254 \times C_F}$ où C_F est le coefficient de frottement des pneus sur la route.

On donne : sur route sèche $C_F = 0.8$ et sur route mouillée $C_F = 0.4$.

Ecrire un algorithme qui donne la distance d'arrêt (en mètres) d'un véhicule quand on connaît sa vitesse en km/h et la météo du jour.

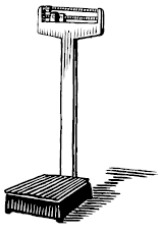
envoyer le dossier algobox à gerard.cordes@ac-nantes.fr avec l'objet : 2Bds12ex2PrénomNom

Classe de seconde : évaluation en salle informatique avec deux autres exos faciles

Activité 6 (IMC) Indice de masse corporelle.

On mesure l'obésité, c'est-à-dire, l'excès de graisse,
à l'aide de l'indice de masse corporelle, noté I,
évalué à partir du poids P (en kg) et de la taille T (en mètres) d'un individu
par la formule: $\mathbf{I = \frac{P}{T^2}}$. I est une fonction des deux variables Pet T.
Suivant une classification établie par l'Organisation Mondiale de la Santé,
un individu est en surpoids lorsque I > 25.

1. a)Calculer l'indice de masse corporelle d'une personne de poids 80kg et de taille 1.75m.
Cette personne est-elle en surpoids ?
b) Même question pour une personne de poids 70kg et de taille 1.70m.



2. a)Ecrire un algorithme qui demande à l'utilisateur son poids en kg et sa taille en m
puis qui calcule l'indice I et enfin qui affiche si l'utilisateur en surpoids ou non.
b) Programmer cet algorithme sur Algobox.

3. Pour un poids de 60 kg, à quelle taille un individu est-il en surpoids?

Testée en classe de seconde

Activité 7 (Feux tricolores)

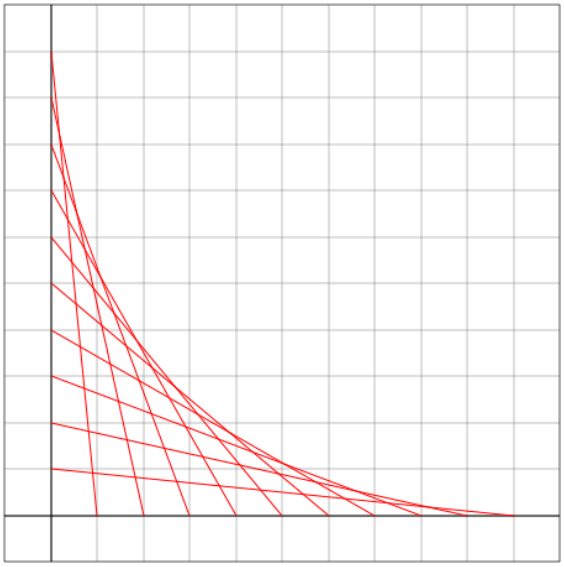
On considère un feu tricolore qui règle la circulation à un carrefour.
Ecrire un algorithme qui détermine la couleur du feu lors du prochain changement connaissant la couleur (vert, orange ou rouge) actuelle du feu.

De même : Aujourd'hui c'est mercredi donc demain c'est jeudi
8h58…8h59…9h…9h01…

Activité 8 (Fils, carrés, escaliers, spirales)

Construire les figures suivantes :

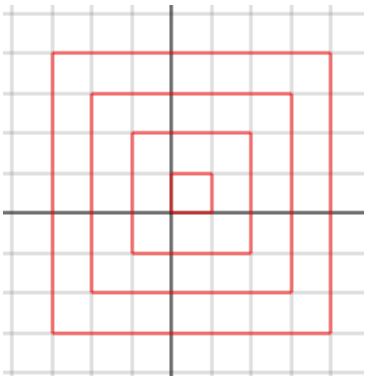
a)Fils tendus



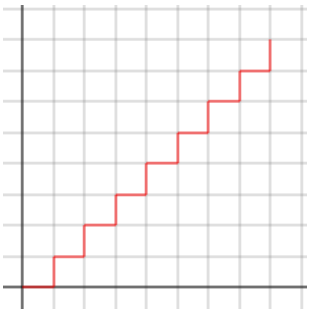
Prolongement :
construction de l'astroïde obtenue par symétrie autour des deux axes

b)Carrés emboités

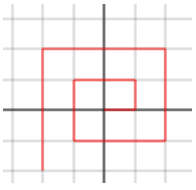
Remarque:
On peut commencer par faire construire un carré de côté c sous algobox.

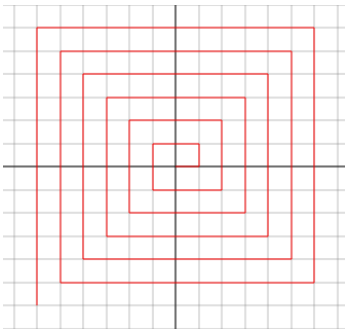


c)Escaliers



d)Spirales

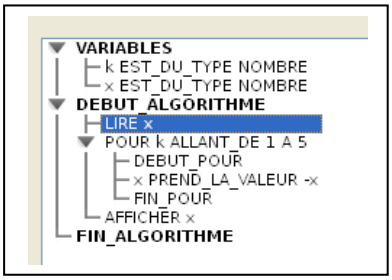




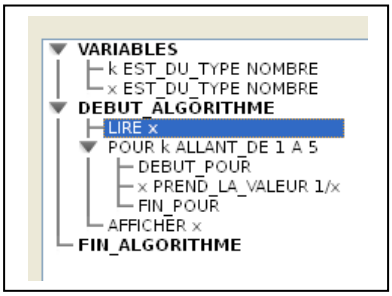
Activité différenciée testée en seconde

Activité 9 (Enchaînements)

1. Voici un algorithme.
a)Que produit cet algorithme ?
b) Que se passe-t-il si on change la ligne « Pour k allant de 1 à 5 » par « Pour k allant de 1 à 6 ». Expliquer.



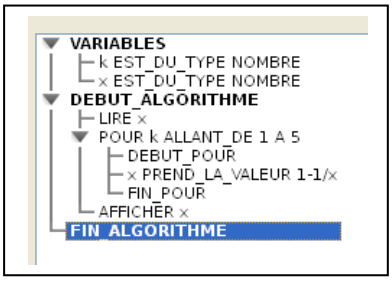
2. Voici un deuxième algorithme.
a)Que produit cet algorithme ? Donne-t-il toujours un résultat ?
b) Modifier cet algorithme pour qu'il donne toujours un résultat.
c)Que se passe-t-il si on change la ligne « Pour k allant de 1 à 5 » par « Pour k allant de 1 à 6 ». Expliquer.



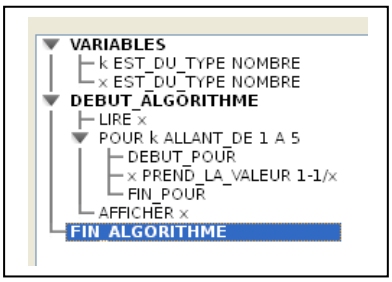
3. Voici un troisième algorithme.

On appelle f la fonction définie sur $]0\ ;\ +\infty[$ par $f(x)=1-\dfrac{1}{x}$.

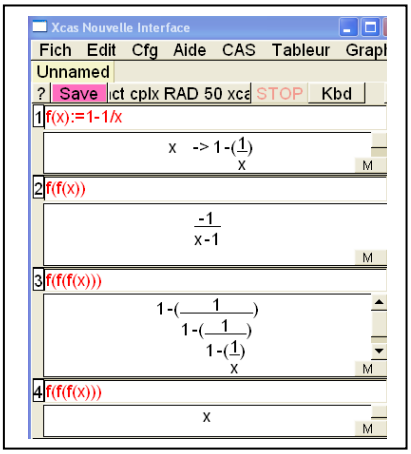
a) Que produit cet algorithme ? Donne-t-il toujours un résultat ?
b) Que se passe-t-il si on change la ligne « Pour k allant de 1 à 5 » par « Pour k allant de 1 à 6 ». Faire une conjecture.
c) Que se passe-t-il si on change la ligne « Pour k allant de 1 à 5 » par « Pour k allant de 1 à 3 ». Faire une conjecture.
d) Prouver la conjecture du 3).
e) Modifier cet algorithme pour qu'il donne toujours un résultat.

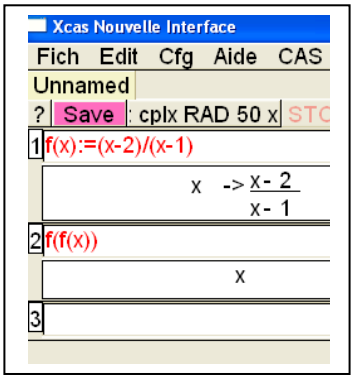


4. Reprendre l'étude du 3) en remplaçant la fonction f par la fonction g définie sur $]0\ ;+\infty[$ par $g(x)=\dfrac{x-2}{x-1}$.



Activité différenciée donnée en AP en seconde(possibilité de faire intervenir le calcul formel)
Pistes:





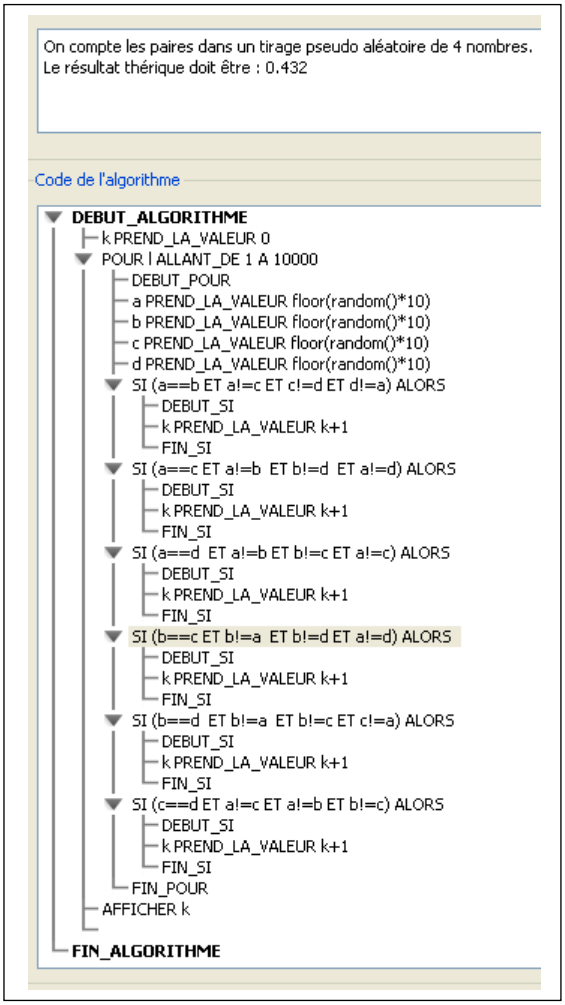
Activité 10 Ma machine est-elle un bon générateur de nombres aléatoires ?

Quand on tire quatre chiffres au hasard entre 0 et 9, on démontre les probabilités suivantes :

Evénements	Probas
Les 4 chiffres sont tous différents	0.504
La liste de 4 chiffres comporte une paire	0.432
La liste de quatre chiffres comporte deux paires	0.027
La lise de 4 chiffres comporte trois chiffres identiques	0.036
La liste de 4 chiffres est formée de 4 chiffres identiques	0.001

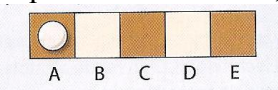
Utiliser ces résultats pour construire un algorithme qui teste si votre ordinateur est un bon générateur de nombres aléatoires.

Activité proposée en travail dirigé en classe de seconde. Il y a eu beaucoup d'intérêt pour cette activité.
On communique aux élèves les résultats du test du poker qui seront démontrés dans les classes suivantes.
L'élèves doit exploiter ces données pour construire un algorithme de son choix qui permette de donner un élément de réponse à la question : « Ma machine est-elle un bon générateur de nombres aléatoires ? »
Voici ci-dessous un exemple de réalisation.



Activité 11 (Déplacement d'un pion)

On dispose d'une rangée de cinq cases nommées A, B, C, D, E. Un pion est placé sur la case A.



Un jeton équilibré porte le chiffre 1 sur l'une de ses faces et le chiffre 0 sur l'autre. On lance ce jeton quatre fois de suite ; lorsqu'on obtient le chiffre 1, le pion avance d'une case vers la droite ; sinon, il reste en place.
On note S la somme des quatre chiffres obtenus.

1. On a construit ci-dessous un algorithme qui simule ce jeu.

• Compléter cet algorithme
• Préciser le rôle de chacune des variables S, *i* et *r*.
• Quel est le rôle de cet algorithme ?

```
Initialisation
   S prend la valeur 0
Traitement
   Pour i de 1 jusqu'à .....
      r prend la valeur 0 ou 1
      S prend la valeur  …
   FinPour
Sortie
   Si S=0 alors
       Afficher « A »
   FinSi

   Si S=1 alors
       Afficher « … »
   FinSi

   Si S= ..... alors
       Afficher « … »
   FinSi

   Si S= ..... alors
       Afficher « … »
   FinSi

   Si S= ..... alors
       Afficher « … »
```

2.a)Modifier l'algorithme précédent pour qu'il simule 1000 fois le jeu et pour qu'il donne la fréquence avec laquelle le pion atteint, à la fin du jeu, chacune des cases A, B, C, D, E.
b) Programmer ce nouvel algorithme sous Algobox et le tester.
c)Conjecturer la probabilité de chacun des événements :
• X : « Le pion reste sur la case A » ;
• Y : « Le pion va au-delà de la case B » ;
• Z : « Le pion atteint la case E ».

3. Prouver les conjectures énoncées au 2.

Activité testée en seconde

Activité 12 (Algorithme de Sébastien)

Voici un algorithme :

LIRE a (entier naturel non nul)
LIRE n (entier naturel non nul)
$1 \to u$
| POUR d =1 à n (pas 1)
| $a \times u \to u$
| FIN POUR
ECRIRE u

a)Faire fonctionner cet algorithme pour $a = 5$ et $n = 3$, $a = 3$ et $n = 5$, $a = 2$ et $n = 10$.
Que semble fournir cet algorithme pour a et n entiers quelconques non nuls?

b) Voici l'expérience qu'a faite Sébastien avec deux nombres secrets distincts x et y entiers naturels non nuls: quand Sébastien lance l'algorithme précédent en saisissant x puis y ou en saisissant y puis x, il obtient le même résultat.
Quels peuvent être les deux nombres secrets de Sébastien ?

Remarques
Cette situation simple a pour but de faire découvrir ce qu'est un algorithme écrit en langage naturel et de le programmer.
Cette activité débouche sur une question mathématique qui intéresse les élèves : quels sont les entiers naturels a et b tels que $a^b = b^a$ *?*
On peut prouver que les nombres secrets de Sébastien ne peuvent être que 2 et 4 en terminale avec la fonction ln .

Activité 13 (Boules et dé)

Un jeu consiste à tirer une boule d'un sac contenant une boule noire et 9 boules blanches toutes indiscernables au toucher, puis à lancer un dé bien équilibré à six faces numérotées de 1 à 6.
Si la boule noire est tirée, il faut obtenir un nombre supérieur ou égal à 4 avec le dé pour gagner.
Si la boule noire n'est pas tirée, il faut obtenir un six avec le dé pour gagner.
1. Calculer la probabilité de gagner à ce jeu.
2. Ecrire un algorithme qui simule une partie de ce jeu et qui affiche, en sortie, le mot « gagné » ou le mot « perdu ».
3. Ecrire un algorithme qui simule 1000 parties de ce jeu et qui affiche en sortie le nombre de fois où le jeu a été gagné.

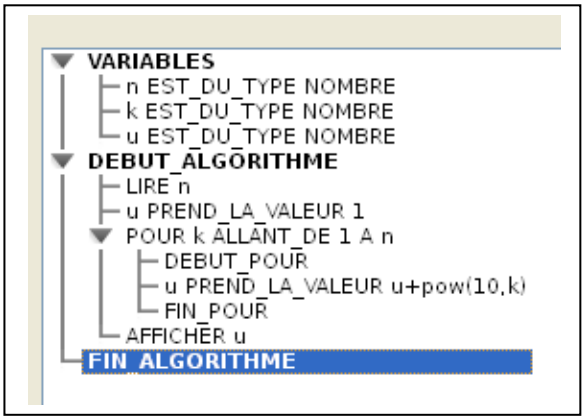
Evaluation en classe de seconde : aucun ordinateur à disposition

Activité 14 (Trois algorithmes pour la classe de seconde)

Première activité

Enoncé

Voici un algorithme programmé avec Algobox



Faire fonctionner cet algorithme avec *n* entier naturel. Que peut-on lire en sortie de cet algorithme ?
Ecrire cet algorithme en langage naturel.

Prolongement ou évaluation

Ecrire en langage naturel et programmer avec Algobox un algorithme qui donne en sortie le nombre entier naturel
121212….12 écrit en base dix avec n tranches « 12 ».

Deuxième activité

Enoncé

1.Voici un premier algorithme :

LIRE un chiffre *x*
LIRE un chiffre y
    u prend pour valeur 10 *x* + *y*
    v prend pour valeur 100 *u*
    w prend pour valeur 100 *v*
    s prend pour valeur *u* + *v* + *w*
Afficher *s*

Faire fonctionner ce premier algorithme.
Comment s'écrit le nombre *s* à la sortie de l'algorithme ?

2. Voici un deuxième algorithme :

LIRE *n* (entier naturel)
u prend pour valeur *n*
    TANT QUE $u \geq 37$
    FAIRE *u* prend pour valeur *u*–37
    FIN TANT QUE
ECRIRE *u*
SI u = 0
ALORS ECRIRE « oui »
SINON Ecrire « non »
FIN SI

Faire fonctionner cet algorithme pour $n = 250$, $n = 185$, $n = 1036$.
Dans quel cas l'algorithme fournit-il la réponse « oui » ? Dans quel cas fournit-il la réponse « non » ?

3. A partir de deux chiffres quelconques x et y saisis dans cet ordre, on lance le premier algorithme qui fournit un nombre entier naturel s. A partir de ce nombre s, on lance le deuxième algorithme.
Faire fonctionner cet enchaînement de deux algorithmes.
Que remarquez-vous ?Quelle conjecture pouvez-vous énoncer ? Prouver cette conjecture.

Troisième activité en guise d'évaluation

On considère l'algorithme suivant.

LIRE n (n entier naturel)
u prend pour valeur $3n^2+3n+6$
Tant que $u \geq 6$
u prend pour valeur $u-6$
Fin tant que
ECRIRE u

a)Faire fonctionner cet algorithme pour plusieurs valeurs de n .
Quelle conjecture pouvez-vous énoncer ?
b)Prouver le résultat conjecturé au a)

Activité 15 Les nénuphars

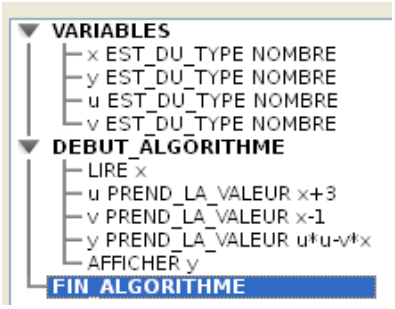
Au pays des plantes géantes, les nénuphars poussent en doublant chaque jour leur surface.
Un matin, un nénuphar éclot au centre d'un étang circulaire d'un rayon de 100m : le nénuphar peut être modélisé par un cercle de rayon 1cm.

1. Déterminer le jour au cours duquel le nénuphar va recouvrir tout l'étang.

2. Deux nénuphars éclosent un certain jour à une distance de 20 mètres l'un de l'autre. L'un mesure 1cm de rayon et l'autre 2cm de rayon. Au bout de combien de jours les feuilles des deux nénuphars vont-elles se chevaucher ?

Activités testée en 1S

Activité 16 Fonction définie par un algorithme



On considère la fonction f définie dans $\mathbf{R}$ en posant : $f(x) = y$
avec y donné par l'algorithme suivant programmé sous Algobox.

1.a)Calculer $f(2)$
b) Calculer $f(10)$
2. La fonction f ainsi définie est-elle affine ?

Activité expérimentée en Accompagement Personnalisé en classe de seconde.

Activité 17 Le plus grand nombre

a)Ecrire un algorithme qui affiche le plus grand des deux nombres x^2 et $\frac{1}{x}$ quand on saisit un nombre *x* en entrée.

b)Ecrire un algorithme qui affiche le plus grand des trois nombres $\sqrt{x}$, x^2 et $\frac{1}{x}$ quand on saisit *x* en entrée.

Activité donnée en 1S en Accompagnement Personnalisé

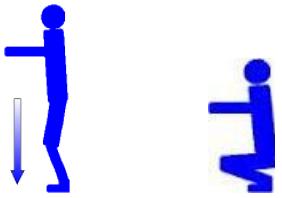
Activité 18 Une suite étrange

On considère la suite (u_n) définie par : $\begin{cases} u_0 = 5 \\ u_{n+1} = u_n + 1 \text{ si } u_n \text{ est pair} \\ u_{n+1} = u_n + 3 \text{ si } u_n \text{ est impair} \end{cases}$

a) Ecrire un algorithme qui donne la valeur du terme de rang *n* quand on saisit l'entier naturel *n*.
b) Ecrire un algorithme qui donne la valeur de la somme des *n* premiers termes quand on saisit l'entier naturel *n*.
c) Ecrire un algorithme qui, à chaque fois qu'on saisit un nombre positif *A*, donne le rang à partir duquel on a : $u_n \geq A$

Utilisation d'un algorithme pour étudier une suite définie par récurrence avec un test pour le passage du rang n au rang n+1 : affichage du terme de rang n, affichage des n premiers termes, étude d'un seuil (cette suie est clairement croissante)

Activité 19 Test de Ruffier



Pour débuter une activité sportive, il est conseillé de passer le test de Ruffier qui permet de calculer l'indice R de résistance de son cœur à l'effort. Pour calculer cet indice, on mesure la fréquence cardiaque (nombre de battements par minute) à trois moments importants pour l'adaptation du cœur :

- Au repos, assis ou couché, on relève la fréquence cardiaque F_1
- Après 30 flexions sur les jambes en 45 secondes, on relève la fréquence cardiaque F_2
- Après une minute de repos, on relève la troisième fréquence F_3.

On calcule ensuite l'indice R par la formule : $R = \frac{(F_1 + F_2 + F_3) - 200}{10}$

a) Ecrire un algorithme qui affiche l'indice de Ruffier en sortie quand on saisit,en entrée, les trois fréquences relevées.
b) En utilisant les données suivantes :

- Si $R \leq 0$, le cœur a une très bonne adaptation à l'effort (cœur « athlétique »)
- Si $0 < R \leq 5$, le cœur a une bonne adaptation à l'effort (cœur « très bon »)
- Si $5 < R \leq 10$, le cœur a une adaptation à l'effort moyenne(cœur « bon »)
- Si $10 < R \leq 15$, le cœur a une adaptation à l'effort insuffisante(cœur « moyen »)
- Si $R > 15$, le cœur a une mauvaise adaptation à l'effort (cœur « faible »)

Ecrire un algorithme qui affiche l'indice de Ruffier et une appréciation en sortie quand on saisit, en entrée, les trois fréquences relevées.
c) Imaginer quelques améliorations de vos algorithmes pour remédier à d'éventuelles erreurs de saisie.

Activité expérimentée en seconde. Pour le c), certains élèves ont proposé de contrôler que : $F_1 < F_2 < F_3$.

Activité 20 Les bonbons de Paul

Paul a 10 bonbons dans sa poche : deux au citron, trois à l'orange et cinq à la framboise.
Paul prend au hasard un bonbon dans sa poche, il le mange, puis il choisit au hasard un deuxième bonbon dans sa poche et il mange ce deuxième bonbon.
On appelle C l'événement « Paul mange deux bonbons au citron »
On appelle O l'événement « Paul mange deux bonbons à l'orange »
On appelle F l'événement « Paul mange deux bonbons à la framboise »
1. Calculer la probabilité des événements : C, O et F.
2. On appelle M l'événement « Paul mange deux bonbons ayant le même parfum».
Calculer la probabilité de l'événement M.

Défi Ecrire un algorithme qui simule 1000 fois cette expérience et qui estime la probabilité de l'événement: « Paul mange deux bonbons ayant le même parfum».

Les deux premières questions doivent être traitées par tous les élèves, le défi étant réservé aux élèves les plus rapides ou les plus motivés.
Voici deux exemples de stratégie pour le défi : dans ces deux algorithmes, les bonbons sont numérotés : 1 et 2 pour le parfum citron ; 3,4 et 5 pour le parfum orange ; 6,7 ,8 ,9 ,10 pour le parfum framboise.
▫ *Stratégie de Melvin : une fois le premier bonbon tiré, il décide d'enlever ce bonbon et de renuméroter les 9 bonbons restants de 1 à 9.*
▫ *Stratégie de Marine : elle programme le tirage d'un deuxième bonbon tant qu'il est n'est pas différent du premier bonbon. On a déjà souligné qu'on n'a pas, en 1S, la preuve que cet algorithme se termine.*

Stratégie de Melvin :

```
VARIABLES
  bonbon EST DU TYPE NOMBRE
  c EST DU TYPE NOMBRE
  o EST DU TYPE NOMBRE
  f EST_DU_TYPE NOMBRE
  k EST_DU_TYPE NOMBRE
  m EST_DU_TYPE NOMBRE
  t EST DU TYPE NOMBRE
DEBUT ALGORITHME
  TANT QUE (k<1000) FAIRE
    DEBUT_TANT_QUE
    bonbon PREND_LA_VALEUR floor(random()*10)+1
    SI (bonbon==1 OU bonbon==2) ALORS
      DEBUT SI
      bonbon PREND LA VALEUR floor(random()*9)+1
      SI (bonbon==1) ALORS
        DEBUT_SI
        c PREND_LA_VALEUR c+1
        FIN SI
      FIN SI
    SI (bonbon==3 ou bonbon==4 ou bonbon==5) ALORS
      DEBUT_SI
      bonbon PREND_LA_VALEUR floor(random()*9)+1
      SI (bonbon==3 ou bonbon==4) ALORS
        DEBUT SI
        o PREND LA VALEUR o+1
        FIN SI
      FIN_SI
    SI (bonbon>6) ALORS
      DEBUT_SI
      bonbon PREND LA VALEUR floor(random()*9)+1
      SI (bonbon>6) ALORS
```

```
33            DEBUT SI
34            f PREND LA VALEUR f+1
35            FIN_SI
36          FIN_SI
37        k PREND_LA_VALEUR k+1
38        bonbon PREND LA VALEUR 0
39        FIN_TANT_QUE
40      t PREND_LA_VALEUR c+o+f
41      AFFICHER "orange "
42      AFFICHER o
43      AFFICHER "citron "
44      AFFICHER c
45      AFFICHER "framboise "
46      AFFICHER f
47      AFFICHER "Même parfum "
48      AFFICHER t
49    FIN ALGORITHME
```

RESULTATS :

```
***Algorithme lancé***
orange 74
citron 22
framboise 197
Même parfum 293
***Algorithme terminé***
```

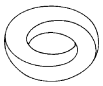
Stratégie de Marine :

```
1     VARIABLES
2       bonbon2 EST_DU_TYPE NOMBRE
3       bonbon1 EST_DU_TYPE NOMBRE
4       d EST_DU_TYPE NOMBRE
5       k EST_DU_TYPE NOMBRE
6     DEBUT_ALGORITHME
7       d PREND_LA_VALEUR 0
8       TANT_QUE (k<1000) FAIRE
9         DEBUT_TANT_QUE
10        bonbon1 PREND_LA_VALEUR floor(random()*10+1)
11        bonbon2 PREND_LA_VALEUR bonbon1
12        TANT_QUE (bonbon2==bonbon1) FAIRE
13          DEBUT_TANT_QUE
14          bonbon2 PREND_LA_VALEUR floor(random()*10+1)
15          FIN_TANT_QUE
16        SI (bonbon1==1 OU bonbon1==2) ALORS
17          DEBUT_SI
18          bonbon1 PREND_LA_VALEUR 1
19          FIN_SI
20        SI (bonbon1==3 OU bonbon1==4 OU bonbon1==5) ALORS
21          DEBUT_SI
22          bonbon1 PREND_LA_VALEUR 2
23          FIN_SI
24        SI (bonbon1>5) ALORS
25          DEBUT_SI
26          bonbon1 PREND_LA_VALEUR 3
27          FIN_SI
28        SI (bonbon2==1 OU bonbon2==2) ALORS
29          DEBUT_SI
30          bonbon2 PREND_LA_VALEUR 1
31          FIN_SI
32        SI (bonbon2==3 OU bonbon2==4 OU bonbon2==5) ALORS
33          DEBUT_SI
34          bonbon2 PREND_LA_VALEUR 2
35          FIN_SI
36        SI (bonbon2>5) ALORS
37          DEBUT_SI
38          bonbon2 PREND_LA_VALEUR 3
39          FIN_SI
40        SI (bonbon1==bonbon2) ALORS
41          DEBUT_SI
42          d PREND_LA_VALEUR d+1
43          FIN_SI
44        k PREND_LA_VALEUR k+1
45        FIN_TANT_QUE
46      AFFICHER d
47    FIN_ALGORITHME
```

RESULTATS :

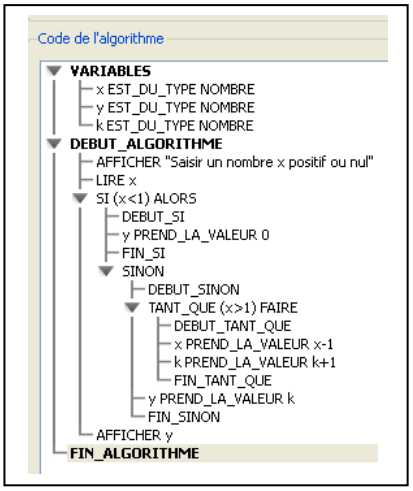
```
***Algorithme lancé***
324
***Algorithme terminé***
```

On peut aussi proposer ces deux derniers algorithmes pour en faire une étude comparative.

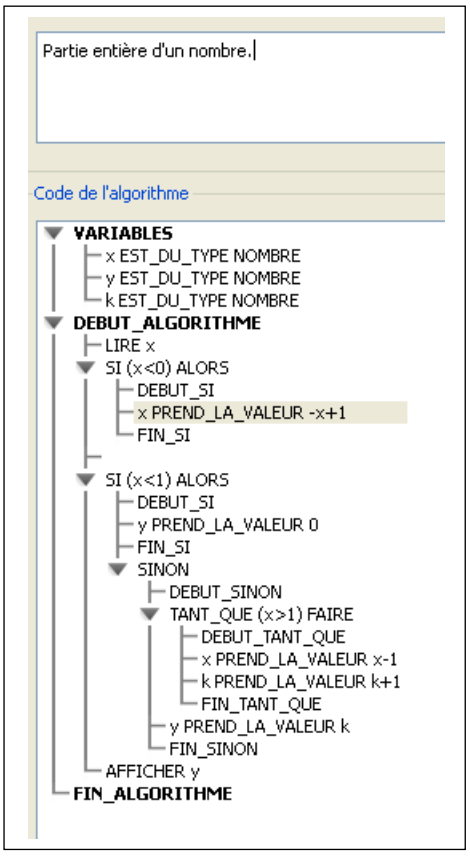


Activité 21 La boîte noire PE

Que produit cet algorithme ?
L'adapter pour qu'il traite aussi le cas où x<0.



Cette situation expérimentée en seconde permet de donner une définition algorithmique d'un outil mathématique : la partie entière.
L'algorithmique permet d'explorer les boîtes noires.



Activité 22 Second degré

Ecrire un algorithme donnant le nombre de solutions strictement positives de l'équation $ax^2+bx+c=0$ quand a, b, et c sont trois réels à saisir avec a non nul.

Activité moins facile que prévue, expérimentée en Accompagnement personnalisé en seconde.
Une bonne occasion de faire le point sur le second degré en prenant une certaine hauteur de vue sur les résultats du cours avec une interaction entre les résultats calculatoires et les résultats graphiques.

Activité 23 La boîte noire !!!

Voici un algorithme :

n désigne un entier naturel non nul à saisir en entrée.
k est un compteur
S est une variable dont le contenu est affiché en sortie

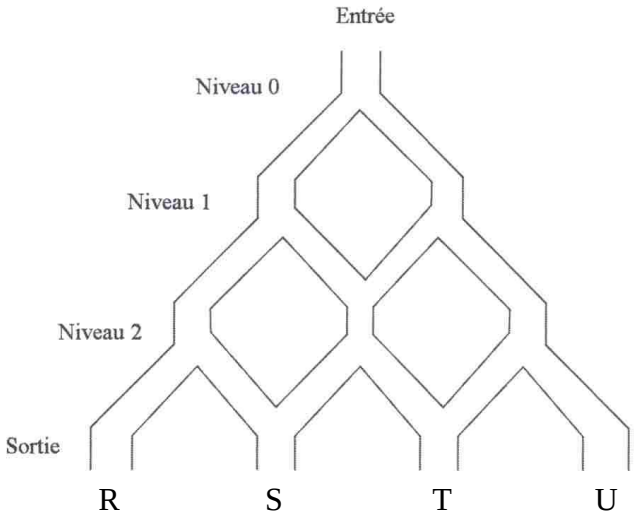
```
1     VARIABLES
2       n EST_DU_TYPE NOMBRE
3       k EST_DU_TYPE NOMBRE
4       S EST_DU_TYPE NOMBRE
5     DEBUT_ALGORITHME
6       LIRE n
7       S PREND_LA_VALEUR 1
8       POUR k ALLANT_DE 1 A n
9         DEBUT_POUR
10        S PREND_LA_VALEUR S*k
11        FIN_POUR
12      AFFICHER S
13    FIN_ALGORITHME
```

Quand on saisit le nombre 12 en entrée on obtient 479001600 en sortie. Vrai ou Faux ? A justifier.

Question donnée au bac blanc2012 en TS. Beaucoup d'élèves ont su répondre et ont justifié. Peu d'élèves ont fait le lien avec la notion de factorielle.

Activité 24 Une souris dans les tuyaux

Une souris descend dans une canalisation (schéma ci-dessous) aboutissant aux sorties R, S, T, U. On suppose que la souris progresse vers l'arrivée en se dirigeant à chaque niveau vers le tuyau de droite ou vers le tuyau de gauche pour accéder au niveau inférieur. La souris ne peut que descendre dans ce réseau de tuyaux.
On s'intéresse à la case de sortie de la souris.



Première partie : simulation
Ecrire un algorithme qui simule 1000 fois la descente de la souris dans ce réseau de canalisations et qui indique la fréquence d'atteindre chacune des quatre sorties.

Deuxième partie :
Compléter le tableau suivant en donnant une preuve des résultats.

Nom de la sortie	R	S	T	U
Probabilité d'atteindre cette sortie				

Travail dirigé en 1S en salle info : deux grandes stratégies se sont dégagées
▫*Travailler avec les abscisses: l'abscisse de la souris est notée xs et se transforme à chaque étape en xs+1 ou xs–1. Avec cette modélisation, on a xt=1, xu=3, xs= –1 et xr= –3.*
▫*Compter le nombre G de déplacements à gauche et le nombre D de déplacements à droite avec D+G=4.*
Voici les algorithmes obtenus :

souris labyrinthe 01

```
1     VARIABLES
2       k EST DU TYPE NOMBRE
3       xs EST DU TYPE NOMBRE (abscisse de la souris)
4       h EST_DU_TYPE NOMBRE
5       N2 EST_DU_TYPE NOMBRE  (abscisse -2)
6       N1 EST DU TYPE NOMBRE  (abscisse -1)
7       P1 EST DU TYPE NOMBRE  (abscisse +1)
8       P2 EST DU TYPE NOMBRE  (abscisse +2)
9       D EST_DU_TYPE NOMBRE
10      N3 EST_DU_TYPE NOMBRE
11      P3 EST_DU_TYPE NOMBRE
12      c EST DU TYPE NOMBRE
13    DEBUT ALGORITHME
14      N3 PREND LA VALEUR 0
15      N2 PREND_LA_VALEUR 0
16      N1 PREND_LA_VALEUR 0
17      D PREND LA VALEUR 0
18      P1 PREND LA VALEUR 0
19      P2 PREND LA VALEUR 0
20      P3 PREND_LA_VALEUR 0
21      POUR c ALLANT_DE 1 A 1000
22        DEBUT_POUR
23        xs PREND LA VALEUR 0
24        POUR k ALLANT DE 1 A 3
25          DEBUT POUR
26          h PREND_LA_VALEUR random()
27          SI (h<0.5) ALORS
28            DEBUT_SI
29            xs PREND LA VALEUR xs-1
30            FIN SI
31            SINON
32              DEBUT_SINON
33              xs PREND_LA_VALEUR xs+1
34              FIN_SINON
35          FIN POUR
36        SI (xs==-3) ALORS
37          DEBUT_SI
38          N3 PREND_LA_VALEUR N3+1
39          FIN_SI
40        SI (xs==-2) ALORS
41          DEBUT SI
42          N2 PREND_LA_VALEUR N2+1
43          FIN_SI
44        SI (xs==-1) ALORS
45          DEBUT_SI
46          N1 PREND LA VALEUR N1+1
47          FIN SI
48        SI (xs==0) ALORS
49          DEBUT_SI
50          D PREND LA VALEUR D+1
51          FIN SI
52        SI (xs==1) ALORS
53          DEBUT SI
54          P1 PREND_LA_VALEUR P1+1
55          FIN_SI
56        SI (xs==2) ALORS
57          DEBUT SI
58          P2 PREND_LA_VALEUR P2+1
```

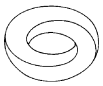
```
59          FIN SI
60        SI (xs==3) ALORS
61          DEBUT_SI
62          P3 PREND_LA_VALEUR P3+1
63          FIN_SI
64        FIN POUR
65      AFFICHER N3
66      AFFICHER N2
67      AFFICHER N1
68      AFFICHER D
69      AFFICHER P1
70      AFFICHER P2
71      AFFICHER P3
72    FIN_ALGORITHME
***Algorithme lancé***
118
377
383
122
***Algorithme terminé***
```

Souris Labyrinthe 02

```
1     VARIABLES
2       c EST_DU_TYPE NOMBRE (compteur du nombre d'expériences)
3       k EST_DU_TYPE NOMBRE (compteur du nombre de niveaux)
4       R EST DU TYPE NOMBRE (nombre de fois où la souris atteint la case R)
5       S EST DU TYPE NOMBRE (nombre de fois où la souris atteint la case S)
6       T EST DU TYPE NOMBRE (nombre de fois où la souris atteint la case T)
7       U EST_DU_TYPE NOMBRE (nombre de fois où la souris atteint la case U)
8       h EST_DU_TYPE NOMBRE (nombre au hasard 0 ou 1 avec0 pour choix d'un tuyau à gauche)
9       G EST_DU_TYPE NOMBRE (nombre de tuyaux choisis à gauche dans une descente)
10    DEBUT ALGORITHME
11      R PREND LA VALEUR 0
12      S PREND LA VALEUR 0
13      T PREND_LA_VALEUR 0
14      G PREND_LA_VALEUR 0
15      POUR c ALLANT_DE 1 A 1000
16        DEBUT POUR
17        G PREND LA VALEUR 0
18        POUR k ALLANT_DE 1 A 3
19          DEBUT_POUR
20          h PREND_LA_VALEUR floor(random()*2)
21          SI (h==0) ALORS
22            DEBUT SI
23            G PREND LA VALEUR G+1
24            FIN_SI
25        FIN_POUR
26        SI (G==0) ALORS
27          DEBUT SI
28          R PREND LA VALEUR R+1
29        FIN SI
30        SI (G==1) ALORS
31          DEBUT_SI
32          S PREND_LA_VALEUR S+1
33        FIN SI
34        SI (G==2) ALORS
35          DEBUT SI
36          T PREND_LA_VALEUR T+1
37        FIN_SI
38        SI (G==3) ALORS
39          DEBUT SI
40          U PREND_LA_VALEUR U+1
41        FIN_SI
42      FIN_POUR
43      AFFICHER R
44      AFFICHER S
45      AFFICHER T
46      AFFICHER U
47    FIN_ALGORITHME
```

RESULTATS :

```
***Algorithme lancé***
114
376
393
117
***Algorithme terminé***
```

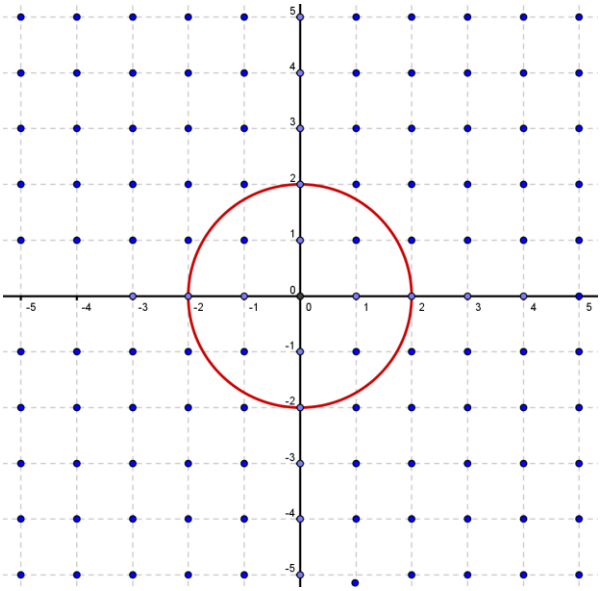



Activité 25 Comptons les points

Séance de travaux dirigés en seconde (pour les questions 1 et 2) ou première S.

Le plan est muni d'un repère orthonormé (O ;I ;J) d'unité 1cm. On se donne un entier naturel *n* non nul.

On se propose compter le nombre S_n de points à coordonnées entières contenus dans le disque de centre O et de rayon $\sqrt{n}$.

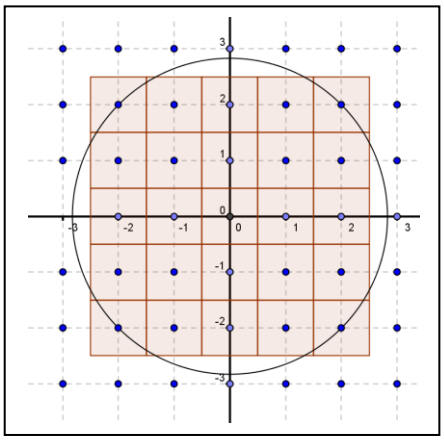


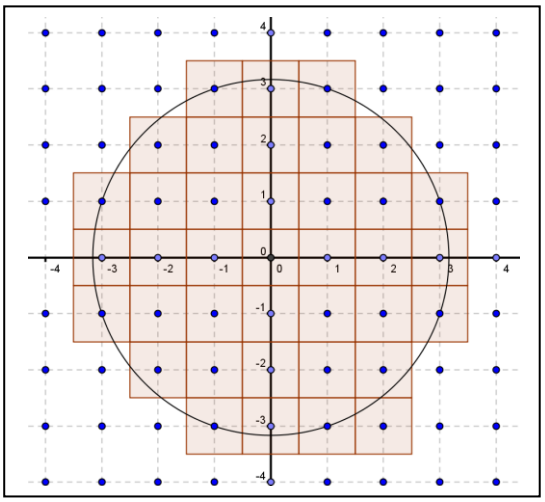
1. Faire le comptage à la main pour *n* égal à 1 , 2 , 3 , 4 , 5.

2. Imaginer une méthode de calcul qui permette de calculer le nombre S_n de points à coordonnées entières contenus dans le disque de centre O et de rayon $\sqrt{n}$.

3. Comment choisir l'entier *n* pour que le disque de centre O et de rayon $\sqrt{n}$ **contienne au moins 1000 points à coordonnées entières ?**

4. Les figures suivantes obtenues pour *n*=8 et *n*=10 permettent de considérer que, pour les grandes valeurs de *n*, l'aire du disque exprimée en cm^2 est voisine de S_n.





Utiliser la méthode de comptage des points précédent pour donner une évaluation grossière du nombre π en choisissant *n* assez grand (par exemple *n*=1000).

Compétences calculatoires :

▫ Mobilisation pertinente de l'outil algorithmique pour effectuer un comptage de points à coordonnées entières dans un disque.
▫ Utilisation du quadrillage pour obtenir une approximation grossière de l'aire d'un disque ou du nombre π.
▫ Suite définie géométriquement dont les termes peuvent être calculés par un procédé algorithmique.
▫ Problème de seuil pour une suite croissante.

Descriptif rapide

▫ Au niveau seconde ou 1S
A partir d'une situation concrète, l'élève découvre qu'un traitement algorithmique est pertinent pour le comptage des points à l'intérieur du disque.
Le professeur intervient pour montrer graphiquement comment un pavage du disque peut conduire à une évaluation grossière de l'aire de ce disque.
En utilisant l'algorithme de comptage de points avec de grandes valeurs de *n*, l'élève arrive à une approximation du nombre π.

▫ Au niveau 1S : les élèves obtiennent une suite originale (Cette suite n'est ni arithmétique, ni géométrique, ni arithmético-géométrique. Cette suite n'est pas définie par récurrence et son terme général n'est pas donné en fonction de *n*). Comme cette suite est croissante, c'est l'occasion de traiter des problèmes de seuil en utilisant à nouveau l'algorithmique.

Scénario :

▫ La situation est vite comprise et le comptage à la main donne :

n	1	2	3	4	5
S	5	9	9	13	21

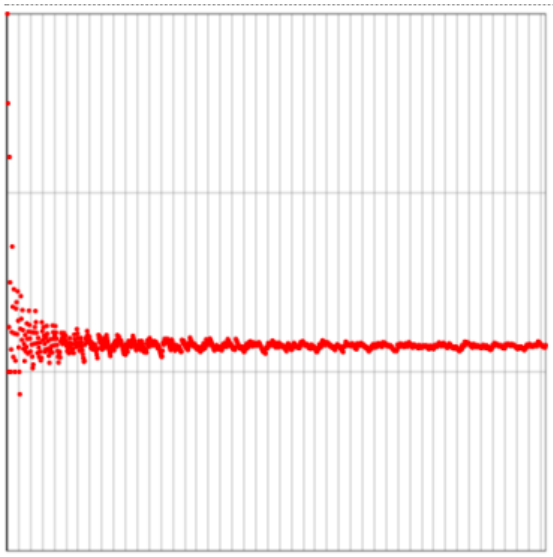
▫ Les élèves ont eu du mal à penser à un algorithme pour effectuer le comptage des points : une fois l'idée amenée par quelques uns, elle s'est vite propagée et elle a été optimisée en considérant le cercle comme quatre quarts de cercle et en testant $x^2+y^2<=n$.
▫ Le problème de seuil a été vite traité par un algorithme (on trouve $n \geq 320$).
▫ La recherche d'une approximation de π a suscité de l'intérêt et des recherches.

Documents

▪ **Algorithme donnant la valeur de S_n en fonction de *n***

```
1     VARIABLES
2       n EST_DU_TYPE NOMBRE
3       k EST_DU_TYPE NOMBRE
4       l EST_DU_TYPE NOMBRE
5       compteur EST_DU_TYPE NOMBRE
6       S EST_DU_TYPE NOMBRE
7     DEBUT_ALGORITHME
8       LIRE n
9       S PREND_LA_VALEUR 0
10      POUR k ALLANT_DE 1 A sqrt(n)
11        DEBUT_POUR
12        POUR l ALLANT_DE 0 A sqrt(n)
13          DEBUT_POUR
14          SI (k*k+l*l<=n) ALORS
15            DEBUT_SI
16            compteur PREND_LA_VALEUR compteur+1
17            FIN_SI
18          FIN_POUR
19        FIN_POUR
20      S PREND_LA_VALEUR 4*compteur+1
21    AFFICHER S
22    FIN_ALGORITHME
```

▪ **Graphique donnant une approximation de π en fonction de n**



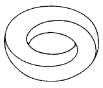
Xmin: 0 ; Xmax: 1000 ; Ymin: 2 ; Ymax: 5 ; GradX: 22 ; GradY: 1

▪ **Algorithme donnant une approximation de π pour n=1000 (question 4)**

```
1     VARIABLES
2       n EST_DU_TYPE NOMBRE
3       k EST_DU_TYPE NOMBRE
4       l EST_DU_TYPE NOMBRE
5       compteur EST_DU_TYPE NOMBRE
6       S EST_DU_TYPE NOMBRE
7       approxpi EST_DU_TYPE NOMBRE
8     DEBUT_ALGORITHME
9       POUR n ALLANT_DE 1 A 1000
10        DEBUT_POUR
11        compteur PREND_LA_VALEUR 0
12        S PREND_LA_VALEUR 0
13        POUR k ALLANT_DE 1 A sqrt(n)
14          DEBUT_POUR
15          POUR l ALLANT_DE 0 A sqrt(n)
16            DEBUT_POUR
17            SI (k*k+l*l<=n) ALORS
18              DEBUT_SI
19              compteur PREND_LA_VALEUR compteur+1
20              FIN_SI
21            FIN_POUR
22          FIN_POUR
23        S PREND_LA_VALEUR 4*compteur+1
24        approxpi PREND_LA_VALEUR S/n
25        TRACER_POINT (n,approxpi)
26        FIN_POUR
27      AFFICHER approxpi
28
29    FIN_ALGORITHME
***Algorithme lancé***
3.149
```

▪ **Algorithme donnant la plus petite valeur de n telle que $S_n \geq A$**

```
1     VARIABLES
2       n EST_DU_TYPE NOMBRE
3       k EST_DU_TYPE NOMBRE
4       l EST_DU_TYPE NOMBRE
5       compteur EST_DU_TYPE NOMBRE
6       S EST_DU_TYPE NOMBRE
7       A EST_DU_TYPE NOMBRE
8     DEBUT_ALGORITHME
9       n PREND_LA_VALEUR 1
10      LIRE A
```

```
11      TANT_QUE (S<A) FAIRE
12        DEBUT_TANT_QUE
13        n PREND_LA_VALEUR n+1
14        S PREND_LA_VALEUR 0
15        compteur PREND_LA_VALEUR 0
16        POUR k ALLANT_DE 1 A sqrt(n)
17          DEBUT_POUR
18          POUR l ALLANT_DE 0 A sqrt(n)
19            DEBUT_POUR
20            SI (k*k+l*l<=n) ALORS
21              DEBUT_SI
22              compteur PREND_LA_VALEUR compteur+1
23              FIN_SI
24            FIN_POUR
25          FIN_POUR
26        S PREND_LA_VALEUR 4*compteur+1
27        FIN_TANT_QUE
28      AFFICHER "Quand n >= "
29      AFFICHER n
30      AFFICHER " le nombre de points est supérieur ou égal à "
31      AFFICHER A
32    FIN_ALGORITHME
```

RESULTATS :

```
***Algorithme lancé***
Quand n >= 320 le nombre de points est supérieur ou égal à  1000
```