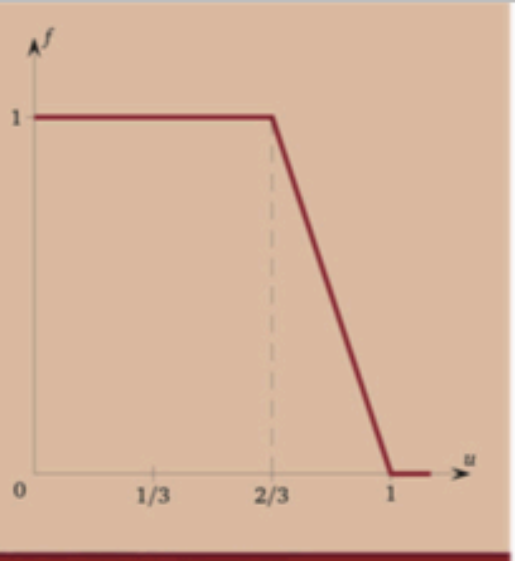



```
image = lire('eden.bmp')

def f(u):
    if u <= 2/3:
        return 1
    elif 2/3 < u < 1:
        return 3*(1 - u)
    else:
        return 0

carmin = Couleur(150, 0, 24)
(xC, yC) = (240, 240)
y = 0
while y < 480:
    x = 0
    while x < 480:
        d = sqrt((x - xC)**2 + (y - yC)**2)
        image[x, y] = image[x, y] * f(d/240) \
                      + carmin * (1 - f(d/240))
        x = x + 1
    y = y + 1

afficher(image)
```

UNIVERSITÉ DE RENNES 1

ENSEIGNEMENT MATHEMATIQUES RECHERCHE ENSEIGNEMENT MATHEMATIQUES RECHERCHE ENSEIGNEMENT MATHEMATIQUES RECHERCHE ENSEIGNEMENT

Introduction de l'algorithmique au lycée

A. Balliot, E. Darrigrand, L. Gaude,
M. Meunier, M. Millet, D. Pinsard

IREM RENNES

Introduction de l'algorithmique au lycée

Brochure IREM – Groupe Algorithmique

Anne Balliot
Éric Darrigrand
Laurence Gaude
Michèle Meunier
Michèle Millet
Denis Pinsard

Plan de la brochure

Listes des activités de la brochure

Introduction

Comme le nom de cette brochure l'indique, nous nous intéressons dans ce document à l'introduction de l'algorithmique au Lycée. Nous avons, la première année, travaillé essentiellement au niveau Seconde, puis la deuxième année nous avons également testé des activités en Première et Terminale.

Il se dégage pour nous aujourd'hui, que la Seconde devrait rester une année d'introduction à des notions algorithmiques, il semble difficile que ces notions puissent être acquises en fin de Seconde. Que peut-on attendre d'un élève en fin de Seconde dans ce domaine ? Les programmes n'abordent pas la question. Une progression de cet apprentissage au cours des années Lycée reste à construire.

Dans la façon dont l'algorithmique apparaît dans le programme de Seconde, grande pourrait être l'impression qu'il ne s'agit que de programmation informatique pour illustrer des concepts mathématiques. *L'art de programmer un algorithme est particulièrement compliqué. Un tel objectif avec des élèves de Lycée requiert un certain nombre de précautions. L'utilisation des outils informatiques ne peut se faire sans apprentissage.*

D'une part, il nous paraît important de garder à l'esprit que nous devons avant tout faire des mathématiques et de ne pas tomber dans l'excès d'un enseignement trop orienté vers l'informatique.

D'autre part, la programmation passe nécessairement par l'utilisation d'une machine programmable. Cela demande un effort d'abstraction qui rend cette activité particulièrement complexe même si parfois certains élèves semblent appréhender l'ordinateur avec beaucoup d'intuition. Il nous paraît important que les élèves soient sensibilisés au fonctionnement d'une machine. Cela contribue à mieux saisir les notions de variables, affectations, et autres concepts spécifiques à l'utilisation d'une machine. La voiture est un moyen de déplacement. Pour l'utiliser, il n'est pas utile de connaître le fonctionnement du moteur dans ses moindres détails mais l'apprentissage de la conduite requiert quelques leçons et quelques connaissances mécaniques (qu'est-ce qu'une boîte de vitesses, un essieu, des plaquettes de frein, ...) Écrire, compiler et exécuter un programme ne nécessite pas de connaître le fonctionnement d'une carte mère mais il est utile d'avoir quelques connaissances de base : gestion de la mémoire (et notion de variables) – processeur de calcul (et notion de séquentialité), …

Cette brochure comporte 2 parties. Les sections 1 à 5 constituent la première partie et décrivent nos réflexions menées sur l'enseignement de l'algorithmique au lycée. La deuxième partie recueille l'ensemble des activités que nous avons testées en classe. Dans la section 1 nous proposons des approches graduées de l'algorithmique :

- graduées en terme d'écriture : Décrire un algorithme / Écrire un programme.
- graduées en terme d'utilisation de moyens informatiques : travail avec ou sans machine programmable.

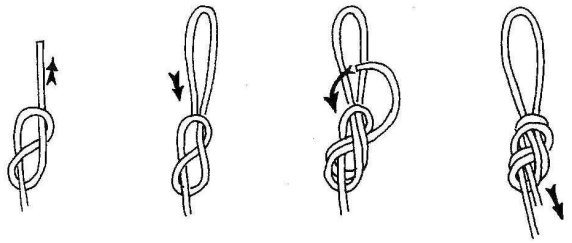
La section 2 interroge sur les objectifs que l'on peut espérer atteindre en fin de Seconde. Dans la section 3, nous discutons de quelques moyens informatiques à notre disposition et proposons une mise en scène théâtrale que nous déclinons sous deux formes. La section 4 traite de l'évaluation. Enfin la section 5 énumère quelques difficultés rencontrées par l'enseignant ou les élèves.

1 Construire un algorithme / Écrire un programme

Dans un premier temps, nous détaillons notre vision de l'algorithmique. En effet, une grande difficulté pour notre groupe a consisté à dégager ce qui pouvait être attendu sous la désignation « algorithme ». D'une certaine façon, chaque utilisation du mot « algorithme » dans le programme de Seconde ne fait référence qu'à l'écriture d'un programme informatique. Dans les ouvrages de Mathématiques de Seconde, rares sont les exercices « algorithmiques » ne faisant pas exclusivement référence à l'écriture d'un programme.

1.1 Définition de « algorithme » – Différence entre algorithme et programmation.

Un algorithme est d'abord une idée, un mode particulier de résolution d'un problème donné. L'apprentissage du calcul posé en primaire illustre bien qu'un algorithme peut ne pas être transcrit.



Cette notion d'algorithme peut être pensée et étudiée en dehors de tout cadre informatique. Comment rechercher efficacement un mot dans un dictionnaire ? Comment sortir d'un labyrinthe ? Les stratégies de calcul mental sont également la source de nombreux algorithmes. Cependant, l'algorithmique et l'informatique entretiennent des liens très privilégiés. Pour peu que l'on ait modélisé et numérisé les données du problème, tout algorithme peut être implanté dans un programme informatique. D'autre part l'informatique engendre elle-même de nombreux problèmes algorithmiques.

Le texte d'un programme possède un rôle double. C'est d'abord une suite de caractères agencés selon une syntaxe bien précise et qui, « interprétée » par la machine, produit des effets particuliers. Le concepteur, mais également le lecteur du programme, doivent être capables de se placer à ce niveau d'interprétation pour comprendre pleinement le texte de ce programme. Mais la signification de ce texte va bien au delà de ce qu'en « comprend » la machine. Ce texte est aussi la transcription de l'algorithme, c'est-à-dire de l'idée, dans un langage codifié. Il ne suffit donc pas que le texte du programme produise les effets voulus, il faut aussi que l'algorithme sous-jacent transparaisse le plus clairement possible. Cela passe en particulier par le choix de noms pertinents pour les variables et par l'insertion de commentaires éclairants.

Dans le cadre des programmes du lycée, les problèmes algorithmiques abordés avec les élèves sont déjà fortement « informatisés » : les données du problème sont des nombres ou des objets numériques directement manipulables par la calculatrice ou le langage de programmation utilisé. Dans ce contexte, la distinction entre l'activité de conception de l'algorithme et celle de conception du programme apparaît

finalement assez vaine. Il faut toutefois ne pas perdre de vue que pour mener à bien cette activité globale il est en particulier nécessaire de savoir se mettre au niveau de la machine. C'est une vraie difficulté pour les élèves qui ne doit pas être occultée.

Ainsi, la notion d'algorithme va bien au delà de la notion de programme. Notre brochure s'articule autour de la considération de ces deux aspects de l'algorithmique : construction et description d'une méthode de résolution – conception et écriture d'un programme.

Dans les activités présentées, nous utiliserons aussi parfois le mot « algorithme » pour désigner un programme.

1.2 Trois niveaux d'écriture d'un algorithme

Afin de mieux saisir les différents aspects de l'algorithmique avec les élèves, il nous a paru intéressant d'introduire une notion d'écriture d'un algorithme sous trois formes :

- La *forme libre* : Il s'agit de décrire une méthode de résolution d'un problème donné sans considération des contraintes liées à la programmation.
- La *forme contrainte* : Si l'algorithme est destiné à être programmé, il est indispensable de pouvoir le décrire par l'utilisation d'instructions programmables (affectations, boucles, tests conditionnels, …). L'écriture sous *forme contrainte* consiste donc à décrire l'algorithme en n'utilisant que des instructions programmables indépendamment d'un langage. L'enseignant ou les élèves peuvent utiliser plusieurs mots pour définir la même action. Deux exemples : « saisir **x**», « lire **x**», « entrer **x** » ou encore « affecter 4 à **x** », « **x** prend la valeur 4 », « mettre 4 dans **x** » ;
- La *forme programmée* : Enfin, pour la mise en œuvre de l'algorithme choisi, le choix d'un logiciel ou d'un langage de programmation s'impose afin de réécrire l'algorithme dans un langage approprié. Ainsi, l'écriture sous *forme programmée* est l'écriture du programme dans un langage de programmation choisi avec toutes les contraintes de syntaxe que cela implique.

En introduisant ces trois formes, nous ne prétendons pas qu'elles doivent être systématiquement présentes ni qu'elles doivent nécessairement se succéder dans l'ordre présenté. La forme libre peut par exemple être rédigée en dernier dans le but de documenter le programme. Le paragraphe suivant précise des articulations possibles entre ces niveaux d'écriture.

Voici quelques exemples d'écriture d'algorithmes sous différentes formes dans le tableau suivant :

Ecrire un algorithme et le programmer - Les différentes phases.

Exemple 1 :	Exemple 2 :	Exemple 3 :
Un magasin de photos propose le développement au tarif de 0.16 €l'unité ; le tarif est de 0.12 €pour une commande d'au moins 75 photos. Ecrire un algorithme qui affiche le prix à payer en fonction du nombre de photos à développer.	Un loyer coûte 720 €et 50 €de charges. Chaque année le loyer augmente de 0.2 % par rapport à l'année précédente et les charges de 3€. Proposer un algorithme qui donne le montant total à payer la douzième année.	On considère une fonction croissante qui possède une unique racine sur un intervalle donné. Proposer un algorithme qui fournit un encadrement de cette racine avec une précision choisie.

1ière phase : On décrit précisément la méthode

1) On demande le nombre de photos à développer. 2) Si ce nombre est inférieur à 75 alors on multiplie par 0.16 et si ce nombre est supérieur à 75 alors on le multiplie par 0.12. 3) On affiche le prix.	1) On multiplie le loyer par 1.002 2) On ajoute 3 aux charges 3) On répète cela 12 fois 4) On fait la somme du loyer et des charges 5) On affiche le résultat	1) En testant le signe de la fonction au milieu de l'intervalle, on détermine de quel côté est située la racine. 2) On sélectionne le demi-intervalle où est située la racine 3) On recommence à l'étape 1 jusqu'à ce que l'amplitude de l'intervalle devienne inférieure à la précision choisie.

2ième phase : On réécrit l'algorithme dans un langage plus codifié

On peut utiliser les « briques » suivantes :

Lire la variable..
On demande quelle valeur affecter à une certaine variable

La variable..prend la valeur..
On affecte une valeur à une variable

Afficher …….
On peut afficher le contenu d'une variable ou afficher un message

Si…alors..

Si…alors ..sinon… On précisera la fin avec FinSi

Il y a deux types de boucles :

Pour …allant de ….à …
Ici **on connait le nombre de boucles** (tours) à réaliser

Tant que……faire…..
Ici, on ne connait pas le nombre de boucles mais on réalise les instructions jusqu'à une condition déterminée. Cette condition s'appelle une **condition d'arrêt**.
On précisera à chaque fois ou se situe la fin de la boucle avec
Fin Boucle

Choix des variables : *N* le nombre de photocopies	Choix des variables : L le prix du loyer C le coût des charges P le prix total i le compteur de boucle	**Données :** • Un intervalle [a, b] • Une fonction f croissante sur [a, b] possédant une unique racine sur cet intervalle. • L'amplitude maximale e de l'encadrement **Résultat :** Un encadrement d'amplitude inférieure à e de l'unique racine de la fonction f. **Traitement :**
Lire la variable *N* Si *N*<75 alors on affiche la valeur 0.16*N* Sinon on affiche la valeur 0.12*N* FinSi	L prend la valeur 720 C prend la valeur 50 Pour i allant de 1 à 12 L prend la valeur L*1.002 C prend la valeur C+3 FinBoucle P prend la valeur L + C Afficher P	**Tant que** b - a > e m = (a+b)/2 **si** f(m) < 0 **alors** a = m **sinon** b = m Afficher a et b

La « rubrique » choix des variables peut se remplir au fur et à mesure de l'écriture de l'algorithme.

Les élèves peuvent écrire facilement l'algorithme de la phase 1, par contre la 2ième phase suppose déjà des connaissances ; il y a les variables à choisir, à initialiser….c'est déjà difficile.
Il y a bien ici des méthodes algorithmiques à enseigner.

3ième phase : On programme cet algorithme

Avec une calculatrice TI	Avec Algobox	Avec Python

PROGRAM:PHOTOS
:Prompt N
:If N<75
:Then
:Disp 0.16*N
:Else
:Disp 0.12*N
:End
	VARIABLES
	L EST_DU_TYPE NOMBRE
	C EST_DU_TYPE NOMBRE
	P EST_DU_TYPE NOMBRE
	i EST_DU_TYPE NOMBRE
	DEBUT_ALGORITHME
	L PREND_LA_VALEUR 720
	C PREND_LA_VALEUR 50
	POUR i ALLANT_DE 1 A 12
	DEBUT_POUR
	L PREND_LA_VALEUR L*1.002
	C PREND_LA_VALEUR C+3
	FIN_POUR
	P PREND_LA_VALEUR L+C
	AFFICHER P
	FIN_ALGORITHME
		def dichotomie(f, a, b, e):
		"""Fournit un encadrement d'amplitude inférieure à e
		de l'unique racine de la fonction f sur l'intervalle [a, b]
		"""
		while b - a > e:
		m = (a+b)/2
		if f(m) < 0: a = m
		else: b = m
		print(a)
		print(b)
		
		def f(x):
		"""Un exemple de fonction"""
		return x**3 - 2*x - 2
		
		# Recherche la racine de f sur l'intervalle [1, 2]
		# au millième près.
		dichotomie(f, 1, 2, 0.001)

On exécute le programme :

prgmPHOTOS N=?60 9.6 Done N=?90 10.8 Done	Résultats ***Algorithme lancé*** 823.47135 ***Algorithme terminé***	1.7685546875 1.76953125

Dans cette phase, on doit apprendre la syntaxe du langage choisi. C'est aussi quelque chose à enseigner.

1.3 Trois approches

Lors de la mise en œuvre d'une méthode de résolution mathématique d'une équation complexe régissant des phénomènes physiques, chimiques, biologiques, il est utile de passer par les trois phases décrites précédemment afin de minimiser les erreurs de programmation et d'en faciliter la correction et la validation qui s'en suit. Évidemment, au lycée, il n'est pas question de résoudre de telles équations. Dans de nombreuses situations très simples, il peut paraître artificiel d'envisager les trois formes d'écriture. Dans d'autres situations, le passage par la forme libre sur papier s'avère plus ou moins inévitable.

Pour résoudre un problème algorithmique, plusieurs approches sont suggérées :

- L'approche *bille en tête* : L'élève ébauche directement une première version de son programme sur une machine programmable sans passer par la phase papier-crayon et l'améliore progressivement par essais et erreurs.
- L'approche *par l'exemple* : L'enseignant peut fournir des algorithmes déjà écrits sous une forme de son choix et faire réfléchir sur le problème résolu ou donner des algorithmes à trous ou suggérer la modification d'un algorithme fourni afin que le résultat en soit modifié selon un objectif précis.
- L'approche *organisée* : À la donnée d'un problème mathématique, on s'efforce de passer par les trois formes d'écriture selon l'ordre dans lequel nous les avons introduites : Tout d'abord, l'algorithme est décrit sous la *forme libre*. Il s'agit d'une étape de réflexion consistant à construire l'algorithme. Ensuite, décrit sous la *forme contrainte*, on obtient une description de l'algorithme n'incluant que des instructions programmables. Enfin, on le programme dans un langage choisi.

1.4 Allers-retours entre les niveaux d'écriture / allers-retours entre les différentes approches

Ces différentes approches permettent une organisation de l'enseignement sur des allers-retours entre les différents niveaux d'écriture d'un algorithme. On peut passer d'une approche à l'autre afin de tirer profit de chacune d'elle. Cela évite aussi de faire paraître la phase papier-crayon comme une tâche scolaire artificielle n'ayant d'autre sens que de satisfaire la “lubie du prof”. On peut combiner ces approches afin d'en tirer les avantages suivants :

- L'approche *par l'exemple* permet d'introduire parfois plus simplement des notions nouvelles. Les élèves peuvent rapidement voir fonctionner un programme. Cette approche peut être privilégiée pour l'élève débutant qui découvre l'activité de programmation d'une machine. Cependant, l'expérience montre qu'il est parfois plus facile d'ébaucher une solution algorithmique à partir de sa propre logique plutôt que de s'immiscer dans la logique d'un algorithme conçu par un autre.
- L'approche *par l'exemple* est aussi une démarche de travail qui consiste à réutiliser en les adaptant des programmes déjà réalisés.
- L'approche *bille en tête* : La démarche par essai et erreur est un moyen efficace pour s'approprier les rudiments de la programmation. Il faut toutefois avoir déjà quelques billes !
- L'approche *bille en tête* peut motiver l'approche *organisée* lorsque les problèmes mathématiques se compliquent.
- L'approche *organisée* est moins motivante que les précédentes. C'est la cerise sur le gâteau. Elle permet d'aborder des problèmes compliqués. L'utilité de cette

approche ne peut être perçue par les élèves qu'après une pratique déjà bien avancée. Sans doute doit-on aborder cette approche dès la Seconde mais ne la mettre réellement en application qu'en Première et Terminale. Elle requiert de la réflexion, de l'organisation. C'est l'approche qui permet de mieux appréhender le concept d'algorithme dans toutes ses dimensions.

Sur la question de la phase papier-crayon, une enseignante du groupe témoigne de ses constats :

*« **J'ai constaté au cours de ces deux années** (l'an dernier une classe très faible, cette année une très bonne classe) **la même chose** :*
C'est le passage sur machine qui les motive, bons ou pas, scolaires ou pas c'est face à la machine qu'ils ont progressé.

Certains élèves qui ne faisaient ordinairement pas grand-chose, se réveillaient lors des séances de programmation ou sur tableur et montraient leur capacité de raisonnement et de logique, se sentaient valorisés.

D'autres, qui avaient des difficultés à l'écrit, pour exprimer leur raisonnement, se sont aussi révélés lors de ces séances et se sont sentis valorisés. Souvent rapides au niveau de la compréhension, ils trouvaient tout de suite ce qu'il fallait faire, même s'ils avaient du mal à passer par la phase écriture de leur algorithme dans un langage qui ne soit pas celui de la machine. Il est arrivé plusieurs fois que je ne comprenne ce qu'ils avaient écrit sur papier qu'en voyant leur programme "machine" ; j'ai trouvé parfois des idées qui étaient plus efficaces que les miennes.

À l'inverse des élèves scolaires pas mauvais du tout, étaient assez bloqués dans ces séances d'écriture d'un algorithme (peur de mal faire, difficultés à essayer des choses, à partir par un bout, revenir à un autre ; ils voulaient une méthode sûre à reproduire, une recette). Sur papier, ils ne démarraient souvent pas ou peu. Par contre quand ils étaient devant algobox, par exemple, ils faisaient des essais (au début un peu n'importe quoi, mais ils osaient face à la machine faire quelque chose, essayaient de comprendre et progressaient).

D'une façon générale, le passage à la recherche papier s'est avéré fructueux surtout après essais sur machine, face à un programme qui ne marchait pas:

« j'ai cherché, j'ai vu que çà ne marchait pas (machine qui évalue : pas le prof) bloqué, j'accepte de revenir au papier pour poser tranquillement mes idées et essayer de trouver mon erreur et surtout j'en vois l'intérêt. »

Aujourd'hui, je pense qu'il faut « motiver » la réflexion papier, mais que c'est via la machine qu'ils en verront l'intérêt.

*Déjà « écrire sur machine » n'est pas facile, ils voient vite les exigences de la rigueur d'écriture. Peut-être est-ce une occasion de réfléchir : **qu'est ce qui ne marche pas « l'écriture » ou « l'algorithme » ?** J'ai rencontré plusieurs fois ce problème avec eux : on avait fait un algo, ils l'avaient programmé et horreur : « çà ne marche pas » : conclusion pour eux : c'est l'algo qui est faux.*

Alors que c'était un problème en général de syntaxe. À chaque fois ce genre de situations a été fructueux, on est toujours revenu à l'algorithme : a-t-on fait une erreur de raisonnement ??? Après recherche : on ne voit pas d'erreur, alors on contrôle l'écriture sur machine, en général le voisin trouve l'erreur de syntaxe et tout le monde est soulagé et content.

On a utilisé sur algobox le mode pas à pas, ou sur TI on a ajouté des « Disp » dans plusieurs endroits pour essayer de voir ce qui se passe, où cela ne va pas. »

2 Ce qui nous paraît réalisable à la fin de la Seconde

Il y a une importante différence, en terme d'écriture, entre la phrase

«Ajouter les *n* premiers entiers naturels »

et l'ensemble d'instructions :

```
Déclarer une variable S;
Initialiser S à 0;
Pour i allant de 1 à n faire
   S prend la valeur S+i;
Fin pour
```

Ceci montre le grand-écart intellectuel que doit faire l'élève pour programmer une méthode de résolution. Ce passage de la forme libre à la forme contrainte nous semble très difficile à réaliser **seul** pour un élève de seconde. Des élèves peuvent facilement exprimer des idées sous *forme libre* sans être capables de les programmer.

Quelques objectifs plus modestes :

- Se familiariser avec les instructions programmables de base : déclaration de variable, initialisation, affectation, boucles, instructions conditionnelles, …
- Savoir trouver sur sa calculatrice les instructions citées ci-dessus.
- Savoir faire tourner à la main un algorithme donné, plus ou moins complexe, pouvant comporter une boucle, un test.
- Pouvoir modifier un algorithme donné, soit pour le rectifier s'il est incorrect, soit pour le transformer afin d'obtenir un autre résultat.
- Pouvoir repérer qu'un problème peut être résolu selon une démarche algorithmique (envisager spontanément l'écriture d'un petit programme pour automatiser une tâche).

Ces objectifs de base consistent à faire comprendre aux élèves ce qu'est un programme et à en reconnaître les principes d'organisation c'est-à-dire à en dégager l'idée d'algorithme. Ces bases sont indispensables mais l'activité proprement mathématique ne débute que lorsque les élèves sont confrontés à des problèmes algorithmiques. Les objectifs de base permettent aux élèves de comprendre ces problèmes mais la tâche de l'enseignant est de donner aux élèves les moyens de les résoudre.

Les élèves trouvent souvent assez facilement une méthode générale pour résoudre un problème donné comme par exemple celui du juste prix. La description de cette méthode peut se traduire sous la *forme libre* évoquée ci-dessus. Pour programmer le juste prix il est nécessaire d'avoir l'idée de la dichotomie et il est nécessaire de savoir aussi ce qu'est un programme. Écrire le programme consiste à faire la jonction entre ces deux choses et cela n'est pas immédiat et ne se résume pas à une histoire d'écriture. Cela nécessite une démarche intellectuelle, un raisonnement. Notre tâche est d'apprendre à l'élève à conduire ce type de raisonnement. Là réside l'activité proprement mathématique.

Voici une démarche assez générale qui nous semble pouvoir être appliquée dans de très nombreuses situations :

• Reconnaître que la méthode va consister à répéter un processus et que l'algorithme va donc comporter une boucle.
• Repérer une situation générale (exemple : on remarque que le nombre cherché appartient à tel intervalle).
• Identifier les instructions de la boucle, qui vont permettre de passer d'une situation générale satisfaite en début de boucle à une nouvelle situation générale en fin de boucle (cette nouvelle situation étant censée faire avancer le Schmilblick).
• Déterminer la condition d'arrêt de la boucle.
• Identifier la situation de départ et donc les instructions d'initialisation de la boucle.

3 Moyens : Stratégies originales (mises en scène théâtrales) – Moyens informatiques – des activités et des fiches

Nous décrivons et commentons ici les outils que nous avons testés ou envisagés. Dans un premier temps, nous proposons une mise en œuvre originale des programmes : la mise en scène théâtrale. Ensuite, nous évoquons les différents logiciels informatiques ou langages de programmation considérés. Enfin, nous décrivons les fiches et activités que le lecteur trouvera en fin de brochure.

3.1 Mise en scène théâtrale

Nous proposons ici deux versions de mise en scène théâtrale :

3.1.1 L'*algorithme humain*

Lors de discussions à propos de nos expériences en classe de l'apprentissage de l'algorithmique, une collègue dans un couloir du lycée explique une de ses méthodes : « les élèves vont au tableau et chacun joue le rôle d'une variable... »

Cette méthode pédagogique consiste à apprendre à « dérouler » un algorithme. Elle nécessite un « **metteur en scène** » et des **acteurs**. Le « metteur en scène » lit l'algorithme à voix haute, instruction par instruction, les « acteurs » sont des élèves **au tableau** qui représentent chacun une des variables utilisées dans l'algorithme, chacun notant au fur et à mesure du déroulement de l'algorithme la valeur prise par « sa » variable. Lors des premières séances, l'enseignant peut jouer le rôle de « metteur en scène ».

Les élèves apprécient beaucoup cette mise en scène, cela permet d'appréhender de nombreuses notions délicates de programmation : qu'est-ce qu'une variable ? Qu'est ce qu'une affectation ? Qui intervient quand ? Qu'est ce que la séquentialité (c'est-à-dire la succession ordonnée d'instructions) ? Qu'est ce qu'une boucle ? Un compteur de boucle ? Quelle est la place mémoire (les limitations informatiques sont semblables aux limitations de place sur le devant du tableau) ?

Lorsqu'on travaille avec Algobox, cela peut permettre de bien comprendre le mode « pas à pas » qu'ils utilisent par exemple pour corriger leur algorithme.

Un exemple de mise en scène théâtrale est proposé dans le compte-rendu relatif à la fiche d'activité sur les nombres parfaits et nombres amicaux de la liste d'activités « En Terminale S-Spécialité ».

3.1.2 La *machine humaine*

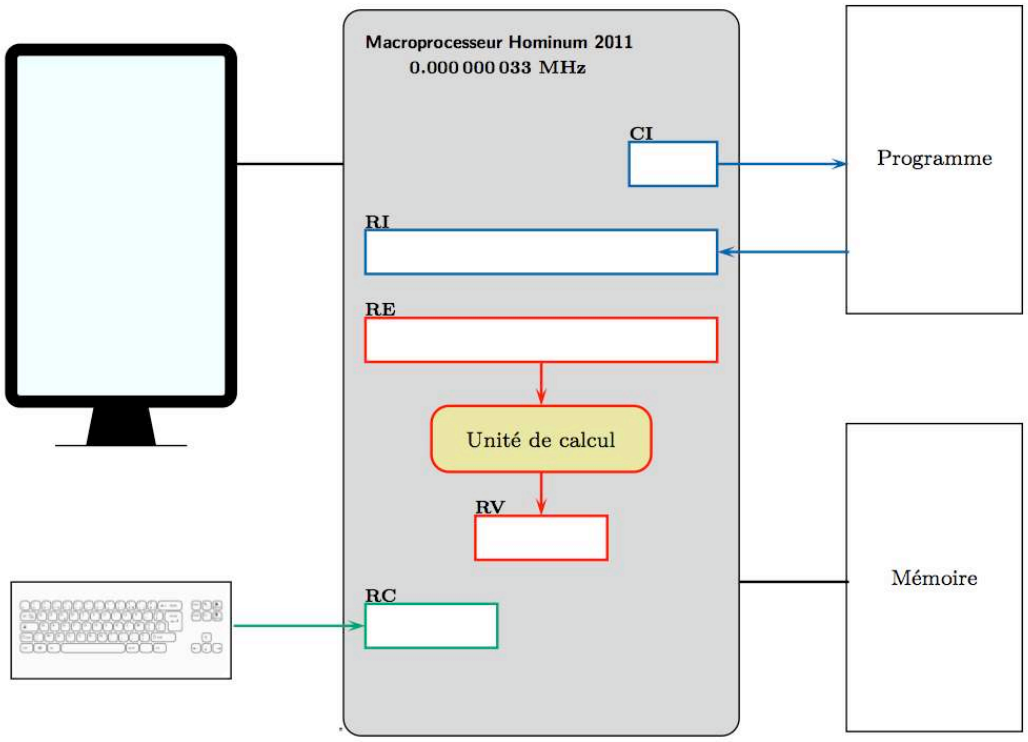
On définit une machine programmable par les entités suivantes : le processeur (il est le chef d'orchestre de la machine), la mémoire (emplacement où sont stockées les variables), l'unité de calcul (le calculateur), …

Expliquer une méthode à un humain et écrire un programme destiné à une machine sont deux activités que l'on peut réunir sous l'expression « écrire un algorithme ». Toutefois ces activités diffèrent autant qu'un humain peut différer d'une machine. Cette dernière exécute mécaniquement des instructions « atomiques » (ou élémentaires), isolées les unes des autres, c'est-à-dire sans aucun lien significatif entre elles. De plus chacune de ces instructions est réalisée par le concours de plusieurs éléments, chacun exécutant une tâche répétitive extrêmement simple. La machine ne prend aucune initiative, elle obéit toujours à la lettre aux instructions, même si les résultats sont aberrants. Tout doit être précisé dans le moindre détail. Si une situation non prévue par le programmeur survient, elle s'arrête ou alors fait n'importe quoi : c'est le bug.

Ces caractéristiques de la machine sont difficiles à percevoir de l'extérieur en raison de la très grande complexité des ordinateurs actuels. C'est pourquoi il nous paraît formateur que les élèves en prennent conscience, ceci d'une part pour démystifier l'ordinateur et d'autre part pour qu'ils comprennent mieux les effets produits par leurs propres programmes. Une façon ludique d'aborder cette question serait de concevoir une mise en scène théâtrale où chaque élève joue le rôle d'un composant d'une machine imaginaire mais au comportement totalement spécifié. Cette machine, c'est-à-dire les élèves, réussirait à exécuter un programme très simple en n'ayant pourtant aucune vision globale de ce programme, et donc en ne sachant pas ce qu'il fait. L'expérience peut aussi montrer la grande difficulté pour un humain de se comporter de façon aussi stupide qu'une machine. Le schéma ci-dessous donne un exemple d'une telle machine humaine. La mise en scène reste à faire ! Tous les détails sont dans le document annexe en fin de brochure.

La *machine humaine* permet de comprendre les raisons des différentes instructions : Pourquoi des variables ? Pourquoi une affectation ? Pourquoi la séquentialité ? Elle donne un sens à tous ces artifices si bizarres au premier abord.

La *machine humaine* a été testée entre nous seulement. Ça fonctionne bien et semble très instructif. Cependant sa mise en œuvre avec des élèves nécessite une bonne connaissance du fonctionnement des machines programmables de la part de l'enseignant.



3.2 Outils informatiques

Au cours des deux années de ce groupe IREM, nous avons testé des activités dont certaines (pas toutes) ont recours à l'outil informatique (calculatrice programmable, tableurs, logiciels de programmation, langages de programmation, …). L'outil informatique ne doit pas être systématique, il est cependant un passage obligé. Comment motiver les élèves à un outil censé permettre des simulations dont le caractère motivant est assez bien partagé par les élèves, et au final leur avouer que l'on ne fera pas de simulation faute d'outil informatique ? Parmi les outils informatiques, nous avons considéré les suivants :

- Calculatrice : C'est l'outil toujours disponible. La programmation y est moins limitée qu'avec AlgoBox que nous présentons ci-dessous. Les programmes écrits sont réutilisables et recyclables. Cependant, la syntaxe reste peu intuitive et rend difficile l'apprentissage de nouvelles instructions programmables. La taille de l'écran limite aussi les possibilités en terme de longueur du programme. Les limites de la vitesse de calcul rendent très longue l'exécution de certains programmes (exemple en simulation).

- Tableur : Il s'agit plus d'une mise en œuvre de programmes inhérents à l'outil que de programmation. Même si on retrouve dans le tableur les notions de variables ou d'itérations, la séquentialité et le traitement algorithmique sous-jacents sont masqués. Tout en étant intéressant, le tableur ne nous semble pas un outil adapté pour enseigner l'algorithmique.

• AlgoBox : AlgoBox est un « logiciel libre et multi-plateforme d'aide à l'élaboration et à l'exécution d'algorithmes (interface graphique de création de pseudo-code algorithmique et d'interprétation du code obtenu en javascript) » par Pascal Brachet *; Licence GPL ;* http://www.xm1math.net/algobox/. Il s'agit d'un logiciel de mise en œuvre d'algorithmes. Il se présente sous forme d'une interface graphique permettant d'écrire son programme par sélections d'instructions dans des menus déroulants. Il offre ainsi la possibilité d'écrire un programme sans avoir à connaître de syntaxe. La syntaxe utilisée, en Français, est assez simple et proche d'une forme que l'on pourrait être tenté de ranger dans la catégorie *forme contrainte* définie dans la section 1. Les adeptes des logiciels à interface graphique devraient l'adopter rapidement. Les élèves sont vite autonomes. Une interface web permet de donner du travail à la maison sur AlgoBox, à condition que les élèves aient un accès internet. Cependant, l'outil présente quelques limitations : Pas de notion de sous-programme ; La syntaxe de l'expression conditionnelle « Si … alors … sinon » est un peu lourde ; La déclaration d'une variable provoque son initialisation à 0 *a priori* (nous conseillons de ne pas tenir compte de cette initialisation à 0 par défaut car ceci n'est pas systématique avec certains langages de programmation avancée) ; Le pas de la boucle « Pour … allant de … à … » est nécessairement égal à 1 ; Comme tout logiciel, AlgoBox requiert une salle TP.

• LOGO : Le langage LOGO a été imaginé il y a déjà plus de 40 ans pour initier les jeunes élèves à la programmation. Le langage possède de réels atouts pédagogiques. Les élèves programment dans un environnement graphique qui permet la réalisation de programmes potentiellement plus motivants que l'exécution automatisée d'une séquence de calculs numériques. Les commandes graphiques sont relatives à la position courante de la tortue. Les élèves se trouvent rapidement confrontés à des questions géométriques relatifs aux angles, aux distances, aux coordonnées. Ce principe de déplacement relatif est tout adapté pour transmettre les idées de vecteurs, de translations, d'angles orientés et en première de calcul différentiel.

Il faut souligner que LOGO est un langage généraliste qui permet de faire bien autre chose que déplacer une tortue dans un plan. Il permet de faire des calculs numériques, de manipuler des mots, des listes de nombres ou de mots, des sons, d'écrire ou de lire dans des fichiers, etc, comme tout autre langage.

Les règles syntaxiques de LOGO sont simples et uniformes. Elles reposent fortement sur la notion de fonction. Ceci peut être utilisé par l'enseignant pour travailler autrement cette notion mathématique, difficile pour les élèves.

Le langage LOGO a été utilisé par l'un des enseignants du groupe en classe de seconde. Cette expérience a été plutôt positive. Elle a permis aux élèves de bien comprendre qu'un programme est un simple texte qui est écrit selon des règles grammaticales précises, c'est-à-dire dans un certain langage, afin d'être correctement interprété par une machine pour produire les effets voulus par le rédacteur. Des outils comme la calculatrice ou Algobox masquent cette idée en encombrant l'activité intellectuelle de construction d'un texte par des tâches techniques de manipulation de touches et de navigation dans des menus.

Cette expérience ne sera toutefois pas renouvelée par l'enseignant qui lui préfère désormais le langage Python dont la syntaxe est simple également et qui a l'avantage de disposer de très nombreuses bibliothèques dans des domaines très variés, dont un module 'turtle' qui permet de retrouver les fonctionnalités de LOGO.

• Python : Le langage Python est un langage de haut niveau dans le sens où il manipule des objets conceptuellement riches, en lien avec les problèmes à résoudre et clairement détachés des mécanismes de bas niveau de la machine. À titre d'exemple, un nombre entier peut avoir une grandeur quelconque.

La syntaxe du langage est particulièrement épurée. Elle privilégie les mots clés en toutes lettres (bien sûr en Anglais) à l'utilisation d'accolades ou d'autres signes cabalistiques. Un programme Python écrit avec soin peut ainsi être tout aussi compréhensible et synthétique qu'un algorithme écrit dans un pseudo-code.

Python est très utilisé par les professionnels dans de nombreux domaines. Il est particulièrement répandu dans le monde scientifique, et possède de nombreuses extensions destinées aux applications numériques.

L'environnement Rurple mérite d'être souligné. Il a été conçu pour initier les élèves à la programmation. Un robot évolue dans un environnement simplifié (polygone orthogonal) et obéit à des ordres élémentaires (avance, tourne à droite, prends une bille, lâche une bille, jette un dé, etc). Le robot peut aussi tester la présence de billes au sol ou dans sa poche ainsi que la présence de murs autour de lui. Les ordres peuvent être donnés individuellement avec les touches du clavier ou programmés en langage Python. Des comportements plus ou moins complexes peuvent ainsi être programmés par les élèves comme par exemple mettre au point un programme qui fait faire au robot le tour de son environnement quelle que soit sa forme.

Python est un langage interprété ce qui facilite le cycle d'écriture et d'exécution d'un programme. Un éditeur de texte tel que Scite permet de réaliser ce cycle de façon simple et conviviale.

Vous pouvez télécharger depuis l'internet à l'adresse
http://dichotomies.fr/2011/infomath/guides/python/installation-python/
un dossier compressé qui contient à la fois l'interpréteur Python et l'éditeur de texte Scite. Vous pourrez également trouver à l'adresse
http://dichotomies.fr/2011/infomath/guides/python/installation-rurple/
un fichier d'installation de l'environnement Rurple pour les systèmes Windows. D'autres documents, guides, activités, verront progressivement le jour sur ce site.

La page http://dichotomies.fr/2011/infomath/python/plan/
liste les documents actuellement disponibles.

• Scilab : Scilab est « le logiciel libre et gratuit de calcul numérique ». Comme tout logiciel de calcul numérique, il requiert la connaissance d'une syntaxe. La documentation, qui a beaucoup évolué ces dernières années est désormais suffisamment aisée d'utilisation pour accéder aux syntaxes des différentes

instructions programmables. Très utilisé dans le domaine de la recherche mathématique, Scilab présente l'avantage que les élèves initiés à ce logiciel investissent durablement lorsque ceux-ci envisagent de poursuivre des études mathématiques après le lycée. Le logiciel est disponible à l'adresse http://www.scilab.org/fr

3.3 Fiches, activités et documents d'accompagnement

La dernière partie de ce document propose une liste d'activités que nous avons pour la plupart testées. Certaines ont été remaniées après leur utilisation. Elles sont rangées selon une organisation qui ne correspond pas nécessairement à une progression chronologique. Parmi ces fiches, un ensemble plus spécifiquement dédié à l'introduction peut facilement servir d'exemples d'appui pour l'enseignant qui ne serait pas à l'aise avec l'enseignement de l'algorithmique.

Certaines des activités proposées en Python ou sur le point de l'être se trouvent à l'adresse :
http://dichotomies.fr/2011/infomath/exercices/2nde/algorithmique/
accompagnées d'un cours introductif
http://dichotomies.fr/2011/infomath/cours/2nde/algorithmique/

Un exemple pour illustrer la démarche algorithmique : L'un des auteurs de cette brochure met à disposition un exemple très largement commenté de mise en œuvre d'un algorithme pour la résolution d'un problème mathématique :
http://dichotomies.fr/2011/infomath/activites/algorithmique/les-deux-verres-doseurs/fiche-prof/

De nombreux autres supports sont aussi disponibles notamment sur les sites internet d'autres IREM. Nous pouvons citer entre autres les ressources suivantes accessibles depuis la rubrique « algorithmique » du portail des IREM (http://www.univ-irem.fr/spip.php?rubrique209) :

- Documents de recherche dédiés à l'initiation des enseignants à l'algorithmique :
  → IREM de la Réunion (http://www.reunion.iufm.fr/recherche/irem) : Fiches d'introduction aux logiciels Scratch et AlgoBox.
  → IREM de Marseille (http://www.irem.univ-mrs.fr) : *Algorithmes et logique au lycée*. Un document qui s'intéresse aux instructions programmables.
  → IREM de Montpellier (http://www.irem.univ-montp2.fr/) : *Introduction à l'algorithmique en classe de Seconde*. Assez technique, décortique des algorithmes particuliers.

- Documents de recherche dédiés à l'approfondissement pour les enseignants :
  → IREM de Marseille (http://www.irem.univ-mrs.fr) : *Une méthode pour élaborer des algorithmes itératifs*.
  → IREM de Lyon (http://math.univ-lyon1.fr/irem) – Fiches d'activités : *algorithmes gloutons* (optimisation locale pour une optimisation globale), *diviser pour régner* (diviser en sous-problèmes), *terminaison d'un algorithme* (aspect validation).

• Documents de recherche dédiés à l'enseignement de l'algorithmique au lycée :
→ IREM de Lille (http://irem.univ-lille1.fr) : *Point de vue sur l'introduction de l'algorithmique en Seconde*.
→ IREM de Grenoble (http://www-irem.ujf-grenoble.fr/irem) : *Document d'accompagnement des stages de formation à l'algorithmique*. Une initiation à la programmation.

• Documents pour la classe : On trouve essentiellement des fiches-activités, des IREM cités ci-dessus ainsi que des IREM
de Reims (http://www.univ-reims.fr/site/laboratoires/irem),
de Franche-Comté (http://www-irem.univ-fcomte.fr),
de Brest (http://irem.math.univ-brest.fr).

4 Évaluation

4.1 BAC : Ce qui existe déjà

En 1L et TL, on enseigne déjà la notion d'algorithme ; elle est évaluée chaque année au bac. Il s'agit en général de comprendre un algorithme donné, de le modifier, voire d'en écrire un autre. Les élèves doivent être capables de « faire tourner à la main l'algorithme » et par exemple de remplir un tableau de données. Ils doivent être capables d'interpréter cet algorithme. Un extrait :

Bac TL Septembre 2009

On considère la suite v de terme général v_n définie par :

$v_0 = 1000$ et pour tout entier n, $v_{n+1} = v_n * 1.005 + 30$.

On considère l'algorithme suivant :

Entrées : Deux nombres entiers S et N
Traitement : Pour K allant de 1 à N
Donner à S la valeur S * 1.005
Afficher : S

Partie A :

Calculer v_1 et donner une valeur arrondie au millième de v_4 .

Faire fonctionner cet algorithme pour S = 1000 et N = 4. Dans l'affichage final arrondir le résultat au millième.

Transformer l'algorithme proposé afin qu'il affiche en sortie finale v_4 .

Partie B :

On place 1000 € sur un livret qui rapporte 0.5 % par mois à intérêts composés. Chaque fin de mois, on y verse la somme de 30 €. Ce livret est bloqué pour 5 ans ce qui signifie que, sur cette période, il est impossible de retirer de l'argent.
Vérifier qu'à la fin du premier mois, la somme présente sur le livret est égale à 1035 €.
Donner un algorithme qui permet d'afficher en sortie finale la somme présente sur ce livret au bout d'une année.

On ne demande pas de calculer cette somme.

4.2 BAC : Avec les nouveaux programmes

En 2nde , 1 et Term, on enseigne maintenant l'algorithmique, il est plus que probable que cela sera évalué au bac dans quasiment toutes les filières. Dans ce cas, les élèves devraient être capables d'interpréter des algorithmes plus ou moins complexes et d'en modifier. Et peut-être, étant donné leur expérience sur 3 ans, être capables d'écrire eux-mêmes un algorithme ; ceci renvoie à de nombreuses questions délicates : Qu'est-ce qu'écrire un algorithme ? Sous quelle forme ? Quelles exigences ? Comment corriger ?

4.3 Évaluation en cours d'année en Seconde

On peut donner dans des DS ou DM un exercice d'algorithmique, relatif ou non au chapitre étudié. Des exemples relativement simples proposés en début d'année :

Algorithme 1	**Algorithme 2**
• Choisir un nombre • Ajouter 2 au nombre choisi • Multiplier le résultat précédent par le nombre choisi au départ • Ajouter 3 au résultat précédent • Annoncer le dernier résultat	• Choisir un nombre • Ajouter 1 au nombre choisi • Mettre au carré • Ajouter 2 au résultat précédent • Annoncer le dernier résultat

Montrer que ces deux algorithmes donnent bien toujours le même résultat quand on entre le même nombre au départ.

Voici un algorithme :

Entrée	Traitement	Sortie
un nombre x	Ajouter 4 multiplier la somme obtenue par x ajouter 2 au résultat précédent afficher le résultat	un nombre y

1) Déterminer y pour $x = 3$.
2) Écrire y en fonction de x.
3) Trouver x pour $y = 2$.

Voici un algorithme :

Entrée	Traitement	Sortie
On lit les coordonnées de 3 points : $A(x_A ; y_A)$ $B(x_B ; y_B)$ $C(x_C ; y_C)$	• On calcule les coordonnées du point I : $x_I = (x_A + x_B)/2$ $y_I = (y_A + y_B)/2$ • On calcule les coordonnées du point D $x_D = 2x_I - x_C$ $y_D = 2y_I - y_C$	On affiche les coordonnées du point D.

Evaluation en algorithmique.pdf

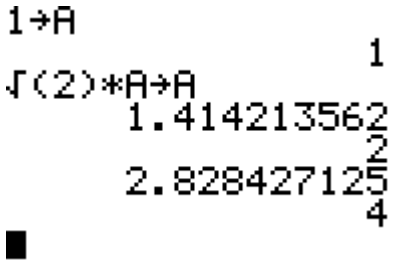
Tester cet algorithme en prenant A (2 ; -1), B (-3 ; 1) et C (5 ; 4).
Faire une figure.
Que fait cet algorithme ? justifier.

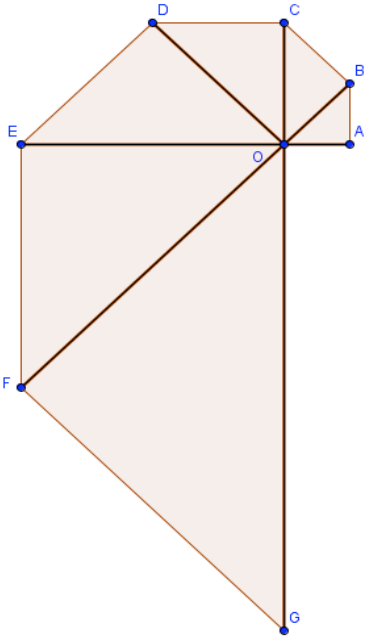
EXERCICE 4

1) Soit un carré de côté de longueur a, Déterminer, en fonction de a, la longueur de sa diagonale.

2) OAB, OBC, OCD, ODE, OEF, OFG sont des triangles rectangles isocèles respectivement en A, B, C, D, E et F. On sait de plus que OA = 1. (Voir la figure ci-dessous)

Expliquer pourquoi à partir de l'écran de calculatrice ci-dessous, on obtient à droite de l'écran, des valeurs approchées des longueurs OA,OB, OC, OD et OE.





On peut aussi demander aux élèves d'écrire un algorithme par exemple avec Algobox et de déposer le fichier sur l'intranet afin de récupérer le travail des élèves.

On peut aussi prévoir ce type d'évaluation en salle info en temps limité, tout comme certains enseignants évaluent des travaux réalisés à l'aide d'un tableur ou d'un logiciel de géométrie dynamique.

4.4 Évaluation en cours d'année en Première et Terminale

On peut évaluer de la même manière avec des exercices plus complexes :

Exemple en 1ES :

Anne, Bruno et Cécile débutent chacun leur emploi avec un salaire mensuel de 1900 €.
Chaque mois, à partir du deuxième, les salaires augmentent ainsi :

- Le salaire d'Anne augmente de 8 €.
- Le salaire de Bruno augmente de 0.4 %.
- Le salaire de Cécile augmente de 0.2 % auxquels s'ajoutent 4 €.

On note A_n , B_n et C_n les salaires respectifs d'Anne, de Bruno et de Cécile le n-ième mois.
On a

$$A_1 = B_1 = C_1 = 1900$$

Salaire annuel

- Calculer le salaire annuel d'Anne pour la première année.

- Calculer le salaire annuel de Bruno pour la première année.
- Voici un algorithme :

```
1    VARIABLES
2      U EST_DU_TYPE NOMBRE
3      S EST_DU_TYPE NOMBRE
4      i EST_DU_TYPE NOMBRE
5    DEBUT_ALGORITHME
6      U PREND_LA_VALEUR 1900
7      S PREND_LA_VALEUR U
8      POUR i ALLANT_DE 1 A 11
9        DEBUT_POUR
10       U PREND_LA_VALEUR 1.004*U+4
11       S PREND_LA_VALEUR S+U
12       FIN_POUR
13     AFFICHER S
14   FIN_ALGORITHME
```

On programme cet algorithme et on obtient :

```
***Algorithme lancé***
23575.9
***Algorithme terminé***
```

Expliquer à quoi correspond ce nombre affiché dans notre contexte, puis ce que représente chaque variable utilisée dans l'algorithme.

Exemple en 1L :

Voir la fiche « Les bonbons » de la rubrique « En Première » avec productions d'élèves.

Exemples en Term S- SpeMath :

Voir la fiche « Évaluation » de la rubrique « En terminale S-Spécialité » : une évaluation en salle info avec le logiciel Scilab.

5 Quelques difficultés rencontrées par l'enseignant ou par les élèves

5.1 Difficultés pour l'enseignant

• **Coller au programme ?** Une lecture un peu rapide des instructions officielles peut donner l'impression que les activités algorithmiques que nous proposons aux élèves doivent systématiquement être en lien avec les autres parties du programme. Ceci ajouterait alors une contrainte pédagogique supplémentaire aux enseignants qui sont en quête des bonnes clés pour transmettre le plus efficacement possible ces nouvelles notions. Une lecture plus attentive des programmes montre que la volonté des auteurs est d'inciter les enseignants à établir des ponts entre les diverses parties, à exploiter au mieux les possibilités d'interactions et surtout à placer cet enseignement dans le cadre de résolutions de problèmes. Mais les instructions officielles n'interdisent pas de s'intéresser à des problèmes algorithmiques autres. Citons : « *L'algorithmique a une place naturelle dans tous les champs des mathématiques et les problèmes posés doivent être en relation avec les autres parties du programme (fonctions, géométrie, statistiques et probabilité, logique)* ***mais aussi avec les autres disciplines ou la vie courante***. »

Ce n'est pas la nature des objets manipulés qui fait de l'algorithmique une activité mathématique, c'est la nature de la démarche qu'elle met en œuvre, du type de raisonnement qu'elle suscite. Intéressons nous donc d'abord aux problèmes algorithmiques pertinents que nous pouvons proposer aux élèves sans se focaliser sur les objets eux-mêmes. Allons s'il le faut les chercher dans les autres disciplines ou dans la vie courante comme nous y invitent les programmes.

• Nous sommes, semble-t-il tous d'accord sur les difficultés relatives aux **dispositions matérielles et techniques** pour un tel enseignement : Les programmes imposent de plus en plus le travail avec les TICE mais les équipements ne suivent pas toujours, il n'est pas forcément facile d'accéder aux salles informatiques très prisées ou d'avoir la possibilité de travailler avec un tableau interactif.

D'autre part, la multiplicité des outils TICE déjà utilisés (logiciels de géométrie dynamique, tableurs, calculatrice, …) pose problème compte tenu de l'horaire imparti et provoque parfois un sentiment de dispersion … De fait, l'ajout d'un logiciel algorithmique peut compliquer davantage l'enseignement des mathématiques.

• **Effectifs** : Une vraie difficulté si on veut faire manipuler les élèves eux-mêmes dans le cas où les classes sont à 35 et dans le cas où des dédoublements ne sont pas possibles.

Des stratégies de contournement (travail en groupe ou autre) ne sont pas toujours faciles à mettre en œuvre.

• Dans le **cas où l'enseignant n'a pas été formé à l'algorithmique** et à la programmation, l'appréhension doit être dépassée, l'auto-formation n'est pas si simple : Certains enseignants n'ont pas eu de formation en algorithmique pendant leur cursus universitaire, ou s'ils en ont eu une, elle remonte à loin et n'a pas toujours été suivie de pratique. Ces enseignants ne sont pas si rares.

Pour un tel enseignant les difficultés sont multiples. Comment enseigner des notions mal maîtrisées, voire mal comprises ? Comment guider les élèves alors que lui-même a des difficultés à s'approprier certaines expressions qui sont loin d'être "naturelles" ? Comment repérer les erreurs faites par les élèves ? Est-il capable de les repérer et d'en cibler le type ?

En d'autres termes, un enseignant qui n'a jamais eu de réelle pratique en algorithmique, et/ou n'a pas d'affinités particulières pour l'algorithmique, va rencontrer les mêmes problèmes de langage que les élèves. S'ajoute à cela, la difficulté de construire une progression sur l'année. Quand on ne maîtrise pas bien un ensemble de notions, comment les articuler de manière cohérente pour les élèves ?

Nous avons distingué au paragraphe 1.2 trois formes d'écriture. La *forme libre* ne pose en général pas de problème à l'enseignant "novice". Le passage de la *forme contrainte* (avec emploi des expressions propres à l'algorithmique) à la *forme programmée* ne pose pas d'autre problème que celui de se plonger dans la syntaxe de l'outil utilisé (calculatrice, logiciel), travail parfois fastidieux mais à sa portée. La difficulté majeure est le passage de la *forme libre* à la *forme contrainte*, tout comme pour les élèves. Un enseignant "novice" comprendra un algorithme écrit sous forme contrainte mais ne sera pas nécessairement capable

d'en écrire un sous cette forme, tout seul, sans "modèle". Et s'il le fait, ne se sentira peut-être pas assez sûr de lui pour le proposer à ses élèves. Le risque alors, est de se contenter d'utiliser les divers exercices qu'offre le manuel choisi par l'établissement, sans bien mesurer les difficultés cachées.

Parmi les fiches de la série d'initiation présentée dans les activités de cette brochure (*Fiches d'initiation : notion de variables et instructions programmables*), certaines ont été remaniées après avoir pris la mesure des difficultés rencontrées, non seulement par les élèves, mais aussi par des enseignants "novices"… Cette série d'initiation devrait permettre à ces enseignants d'aborder plus sereinement les choses. Au bout d'un certain temps et avec plus d'expérience, ils pourront se détacher de ce type de fiches et se lancer dans des choses plus ouvertes.

5.2 Difficultés pour les élèves

• En algorithmique, la notion de variable est intimement liée au concept de mémoire machine et au besoin d'une case mémoire pour stocker la valeur de telle ou telle quantité. Il faudra veiller à ne pas sous-estimer les difficultés des élèves à appréhender cette notion qui est loin d'être naturelle.

• La notion de boucle « Pour **i** allant de … à … » est aussi un concept compliqué à acquérir. Dans un premier temps les élèves préfèrent le « Répéter **n** fois ». La notion de compteur associé à ces boucles est souvent l'origine de difficultés. Le « Tant que … faire ... » et le « Répéter … jusqu'à … » leur sont plus naturels cependant la condition d'arrêt qu'ils impliquent reste une notion délicate.

• Les notions de place mémoire requise et de coût calcul d'un algorithme sont essentielles et il semble primordial d'aborder ces questions dès le lycée. Cependant, il nous paraîtrait prématuré de vouloir initier les élèves de Seconde à l'importance de ces considérations.

Deux concepts peuvent être source de difficultés et sont parfois mal appréhendés par les enseignants eux-mêmes :

• Le test d'égalité et la précision machine : La précision machine désigne le zéro machine. Informatiquement parlant, tout nombre plus petit que la précision machine sera considéré comme nul devant l'unité. Par exemple, si la précision machine est de l'ordre de 10^{-16} , $1+10^{-17}$ vaudra 1 pour la machine. Par contre, le nombre $0.0004-10^{-17}$ est bien différent de $0.0004+10^{-18}$ pour cette machine alors que ces deux quantités peuvent correspondre à deux variables qui valent le même nombre (ici 0.0004) pour l'utilisateur. Ceci peut arriver lorsque les nombres supposés par l'utilisateur sont calculés par la machine au cours d'un programme. Ainsi, ces deux nombres sont égaux pour l'utilisateur et différents pour la machine. De fait, un test d'égalité entre deux réels **a** et **b** doit se faire par évaluation de la différence en valeur absolue entre ceux-ci et non par test d'égalité : les deux réels sont considérés identiques si |**a**-**b**| est plus petit qu'une précision donnée que l'on choisit un peu plus grande que la précision machine, par exemple 10^{-14} dans le cas de notre exemple où la précision est de l'ordre de 10^{-16}. Un test du type « |**a**-**b**| = = 0 » ou « **a** = = **b** » est une erreur de program-

mation pouvant induire par exemple des divisions par zéro.

Un test simple pour vérifier la précision machine :

a = 10^{-14}

a = a + 1

a = a - 1

Si à la fin de ce test, **a** a toujours la même valeur, alors la précision machine est plus petite que **a**. On peut répéter le test avec d'autres puissances de 10. Les précisions machine habituelles sont 10^{-8}, 10^{-16} ou 10^{-32}.

• « Si …, si ... » ou « Si … sinon » : Rares sont les élèves qui ont le réflexe d'utiliser le « sinon ». Il leur est plus facile de proposer une liste de « si ». Mais alors pourquoi utiliser le « sinon » ? Cette fois-ci, il ne s'agit pas d'erreur de programmation mais de coût d'évaluation d'une condition. Lorsque l'on écrit « Si `condition` alors `instruction1` sinon `instruction2` », la machine ne doit évaluer qu'une condition, même dans le cas où c'est l'instruction 2 qui va être effectuée. Lorsque l'on écrit « Si `condition` alors `instruction1` , si `contraire de condition` alors `instruction2` », dans le cas où c'est l'instruction 2 qui doit être réalisée, il y aura deux évaluations de condition au lieu d'une seule. Parfois, les évaluations des conditions sont plus coûteuses que les instructions elles-mêmes. Dans ces cas, l'utilisation du « sinon » est primordiale.

Nous terminons ce paragraphe en revenant sur l'importance de varier le vocabulaire selon lequel les algorithmes sont énoncés. Ceci rejoint les réflexions menées sur les différentes formes d'écriture d'un algorithme ainsi que sur les différentes approches définies en début de document. Une enseignante du groupe témoigne sur les difficultés des élèves à appliquer un algorithme par non compréhension du vocabulaire utilisé dans l'énoncé :

« *L'algorithme de coloriage des sommets d'un graphe et l'algorithme pour trouver un plus court chemin sont en général présentés dans les manuels de TES (spécialité) sous la forme "Tant que … faire …". Sous cette forme les élèves ont beaucoup de mal à comprendre ce qu'il faut faire en réalité, notamment pour l'algorithme du plus court chemin (plus complexe que l'algorithme de coloriage car il comporte un "si … alors").*

Pour que les élèves comprennent, il faut en général revenir à une description plus proche de la réalité de ce qui se passe dans nos petites têtes d'humains.

Lorsqu'on présente ces algorithmes sous l'une des deux formes ci-dessous, les élèves comprennent nettement mieux ce qu'ils doivent faire réellement.

① `Faire …`
② `Répéter/recommencer l'étape précédente jusqu'à ce que …`

Ou, plus directif

① `On fait …`
② `On recommence …`
③ `On arrête quand … ou dès que … »`

Les activités de la brochure

En classe de seconde

Activités sans prérequis d'ordre algorithmique

- Construction de nombres rationnels
- Algorithme de Babylone
- Le nombre d'or (*Calculatrice*)
- Léo et Léa
- Résolution géométrique d'équation (Al-Khwarizmi)
- Évaluation d'expressions numériques
- Robots-tiroirs

Construction de nombres rationnels

Activité très connue qui peut être faite en début d année de seconde, l idée est ici de proposer une solution de type algorithmique à un problème posé, de montrer que la résolution algorithmique dun problème ne date pas d aujourd hui.

Plusieurs façons d aborder la question, en voici deux exemples :

Sous la forme d une question ouverte du genre :

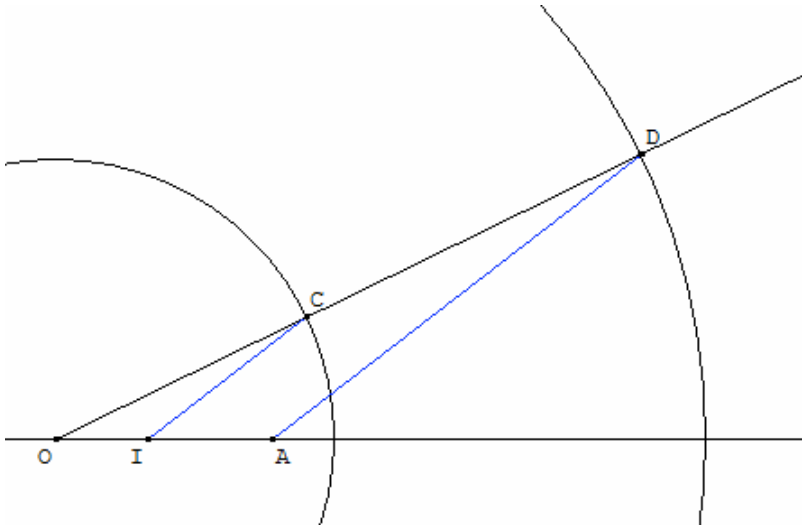
« Un axe (O, I) étant donné, trouver une construction à la règle et au compas qui permet de déterminer le point P de l axe (O, I) tel que $OP = \frac{7}{3}$ »

Autre :

On donne la figure suivante, les arcs de cercles sont de centre O l un passe par C l autre par D, les points O, C, D sont alignés ainsi que les points O, I, A; de plus, les segments [IC] et [AD] sont parallèles.
OI est prise pour longueur unité c'est-à-dire que l on pose OI = 1

a) OC = 3 et OD = 7. Calculez OA.
b) Construire sur [O I) le point B tel que $OB = \frac{11}{7}$ puis, le point E tel que $OE = \frac{2}{3}$, à la règle (non graduée) et au compas.
c) Décrire une construction, qui permet de construire un segment de longueur $\frac{a}{b}$ où a et b sont des entiers naturels non nuls, à la règle (non graduée) et au compas.

.

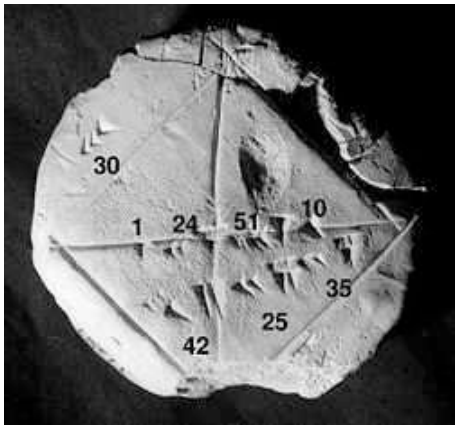


Prolongement :

Construire un segment de longueur $\frac{a}{b}$ où a et b sont des entiers (b non nul) à l aide d un logiciel de géométrie dynamique, a pouvant varier de 0 à 15 et b de 1 à 15.

Algorithme de Babylone

I) Mathématiques babyloniennes (d'après Wikipédia)



Photographie de la tablette YBC 7289 annotée.

Les nombres écrits dans le système babylonien donnent une approximation de la racine de 2 sous la forme 1 24 51 10, en sexagésimal, c'est-à-dire, en décimal : 1, 414 212 963, au lieu de 1, 414 213 562. Pas si mal !

$1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3} = 1{,}41421296$

Les **mathématiques babyloniennes** sont les mathématiques pratiquées par les peuples de l'ancienne Mésopotamie (dans l'Irak actuel), depuis l'époque des Sumériens jusqu'à la chute de Babylone en 539 av.J.Chr. Alors que l'on ne dispose que de très rares sources sur les mathématiques en Égypte antique, notre connaissance des mathématiques Babyloniennes s'appuie sur environ 400 tablettes d'argile mises au jour depuis les années 1850. Écrites en cunéiforme, ces tablettes furent travaillées sur de l'argile encore humide, puis cuites dans un four ou séchées au soleil. La plupart des tablettes qui nous sont parvenues datent de 1800 à 1600 av. J. C, et traitent de fractions, d'équations algébriques (équations du second degré et du troisième degré), de calculs d'hypoténuse et de triplets pythagoriciens voire, peut-être, de certaines lignes trigonométriques (cf. notamment la tablette Plimpton 322). La tablette YBC 7289 fournit une approximation de $\sqrt{2}$ précise à six décimales près.

Les historiens des mathématiques se sont interrogés pour savoir comment les Babyloniens avaient obtenu cette excellente approximation. Voici une réponse possible :

II) Calcul d'une approximation de $\sqrt{2}$

1- $\sqrt{2}$ est le côté d'un carré dont l'aire est 2, comment construire un tel carré ?

L'idée est de s'en approcher pas à pas de la façon suivante :

On part d'un rectangle de côtés respectifs 1 et 2 son aire est donc 2 .

On va rendre ce rectangle plus « trapu » en construisant un autre rectangle .

2- Pour cela :

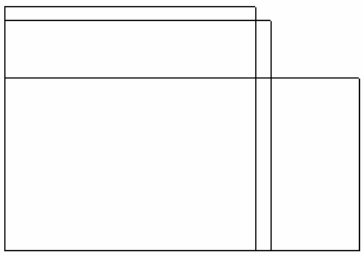
On choisit pour longueur la moyenne des dimensions du précédent rectangle : L =

puis on choisit la largeur de façon à ce que l'aire soit encore 2 ; Calculez cette largeur : l =

Le nombre que l'on cherche à savoir $\sqrt{2}$ est compris entre et

Puis, on recommence autant de fois qu'on le souhaite.

Cette construction peut être faite à la règle et au compas sans calcul : voir III, on obtient au bout de trois constructions :



3- Convergence : Observation

Géométriquement, on a envie de dire que la figure tend vers un carré et que le côté tend vers la racine carrée de 2.

Or les côtés sont exprimés de proche en proche, à partir de 1 et 2 par de simples additions et divisions. On peut donc obtenir par le calcul des approximations fractionnaires de la racine de 2 aussi proche qu'on le désire.

On suppose que c'est ce qu'on fait les Babyloniens.

A ma connaissance, on ne trouve cependant pas un exposé de cette méthode dans les tablettes babyloniennes. On le trouve, beaucoup plus tard, chez Héron d'Alexandrie (62 après JC).

4- A vous : on décide d'appeler l_0 la largeur du rectangle initial soit $l_0 = 1$ et L_0 la longueur du rectangle initial soit $L_0 = 2$ et ainsi de suite on aura $l_1 = \frac{4}{3}$ et $L1 = \frac{3}{2}$.

Complétez le tableau

étapes	longueur	largeur	encadrement de $\sqrt{2}$ par des fractions	encadrement de $\sqrt{2}$ par des décimaux
initiale	$L_0 = 2$	$l_0 = 1$	$1 < \sqrt{2} < 2$	
1	$L_1 = \frac{l_0 + L_0}{2} = \frac{3}{2}$	$l_1 = \frac{2}{L1} = \frac{4}{3}$	$\frac{4}{3} < \sqrt{2} < \frac{3}{2}$	$1{,}33 < \sqrt{2} < 1{,}5$
2				
3				
4				

III) Une construction.

1) On donne OACB un rectangle tel que OA = 2 et OB= 1.
   Construire :
   - Le point I_1 intersection du cercle de centre O de rayon OB avec la demi-droite [OA).
   - A_1 le milieu du segment $[I_1 A]$.
   - La droite (A_1B).
   - La parallèle à la droite (A_1B) passant par A.
   - Le point B_1 intersection de cette parallèle avec la droite (OB).
   - Le point C_1 intersection de la perpendiculaire à (OA_1) passant par A_1 et de la perpendiculaire à (OB_1) passant par B_1.

   Résultat : Le rectangle $OA_1C_1B_1$.

2) Justifier que $OA_1C_1B_1$ est bien un rectangle.

3) Démontrez que l'aire de $OA_1C_1B_1$ est la même que celle du rectangle de départ OACB.

4) Reprendre les constructions de la question 2) en partant non plus de OACB mais de $OA_1C_1B_1$ (Vous remplacerez tous les 1 par des 2. Que dire de l'aire du rectangle $OA_2C_2B_2$?

5) Recommencez encore avec le rectangle $OA_2C_2B_2$ pour rectangle de départ, vous obtenez le rectangle $OA_3C_3B_3$ que vaut l'aire de ce dernier rectangle ?

Compte rendu : activité algorithme de Babylone

Sources : APMEP bulletin n° 486 ; article de Wikipédia ; équipe académique de Bordeaux décembre 2002

Pré requis : Des notions de collège : calcul sur les fractions, théorème de Thalès

Conditions d expérimentation et durée: Classe de seconde, une séance en Travaux dirigés (prolongement en travail personnel), tout début d année scolaire, salle équipée d un vidéo- projecteur.

Objectif : Mettre en place la notion d algorithme, montrez l intérêt d un outil de programmation.

Déroulement :

Je distribue la page 1 et les laisse lire le I et commencerle II, je commente le I en précisant que les changements de bases ne sont pas l objectif de la séance.
J explique oralement avec eux la méthode décrite dans le II
Je sens que ce serait bien de justifier l encadrement, ce serait un prolongement possible àcette activité dans une bonne classe, mais dans ce cas une séance ne suffirait pas
Ils complètent sans trop de problèmes L= et l = (quelques erreurs pour l)
J explique que j ai fait la construction à l aide d un logiciel de géométrie dynamique,la construction est décrite au III, ils la feront à la maison seuls pour la prochaine fois (lunité : 5 cm)
Là encore il serait intéressant de leur faire faire la construction lors d une autre séance avec par exemple géogébra en utilisant le menu « outils » pour refaire le tracé du nouveau rectangle de façon automatique.

On passe à l idée de convergence qu ils acceptent sans problème, j évoque la question : obtiendra t-on un carré au bout de 100 répétitions de tracés??? Pourquoi ??? Là forcément, c est trop difficile, mais tous pensent qu on ne va jamais s arrêter.

Ensuite, ils complètent le tableau. Quelques uns n ont pas compris tout de suite qu il fallait repartir des dernières valeurs calculées mais à la fin de la 2e c était compris.
Comme on avait refait pas mal de calcul mental sur les fractions et que la classe calculait plutôbien, je leur ai demandé d utiliser leur calculatrice pour faire les calculs: utilisation **de FRAC sur TI** : on est allé chercher dans le menu .
Pour la ligne 4 : Il faut faire à la main , pas assez d espace sur l écran pour la fraction, les approximations données sur l écran sont les mêmes, on est allé**chercher les 3 dernières décimales stockées mais qui napparaissent pas** (* 1000-1414) : génial !!! M ont dit certains, est ce qu il y en a d autres de cachées ..
On a eu aussi l occasion de parler de**troncature, d arrondis au moment des encadrements**demandés en dernière colonne

A ce stade, il reste 6-7 minutes : j ai fait trop de digressions.
Je montre le programme fait sur algobox, ils reconnaissent, ce qu ils ont fait à la main.
Je demande combien de fois le calcul sera refait; je le fais fonctionner, en 10 s les résultats apparaissent .
Manque de temps, je distribue l algorithme, leur dit qu ils peuvent télécharger algobox : logiciel Gratuit et essayer chez eux, ils restent un peu sur leur faim certains auraient voulu reparler de la précision affichée.

Bilan : Activité riche,
j aurais dû aller plus vite sur la partie historique
pour **garder plus de temps pour algobox**
qui était en fait mon réel objectif.

approximation de racine de 2

```
1   VARIABLES
2     L EST_DU_TYPE NOMBRE
3     la EST_DU_TYPE NOMBRE
4   DEBUT_ALGORITHME
5     L PREND_LA_VALEUR 2
6     la PREND_LA_VALEUR 1
7     POUR I ALLANT_DE 1 A 10
8       DEBUT_POUR
9       L PREND_LA_VALEUR (L+la)/2
10      la PREND_LA_VALEUR 2/L
11      AFFICHER "rac(2) est compris entre  "
12      AFFICHER la
13      AFFICHER "  et  "
14      AFFICHER L
15      FIN_POUR
16  FIN_ALGORITHME
```

Nombre d'or – Approximation

A partir d'une activité proposée sur le site de G Connan.

Partie I : Le nombre d'or

1) Soit ABCD un carré de côté 1. Retrouver les étapes de la construction du rectangle ADFE, puis refaire la construction sur votre copie en choisissant pour unité 10 cm. (C et E sont sur un cercle de centre I, avec I milieu de [AB]).

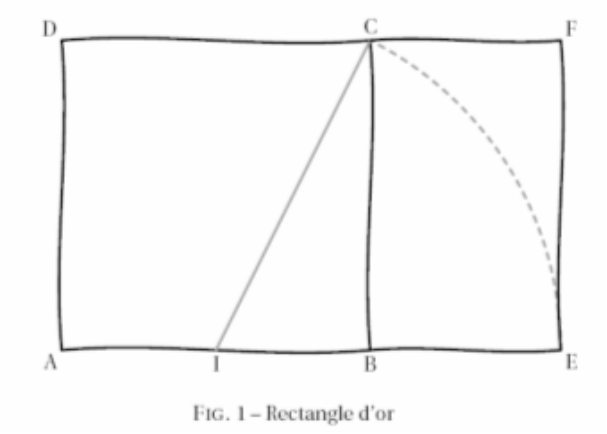


FIG. 1 – Rectangle d'or

2) Montrer que $AE = \dfrac{\sqrt{5}+1}{2}$. Ce nombre est noté ϕ et l'on l'appelle le nombre d'or. Le rectangle obtenu par la construction donnée est appelé un rectangle d'or.

3) Donner, à l'aide de votre dessin, une valeur approchée de ϕ. Quelle est à votre avis l'ordre de grandeur de la précision ?

4) A l'aide de la calculatrice, donner une valeur arrondie à 10^{-5} près de ϕ.

Partie II : Des suites de nombres…

1) A l'aide de la calculatrice, donner une approximation des nombres suivants :

$a_1 = \sqrt{1+\sqrt{1}}$, $a_2 = \sqrt{1+\sqrt{1+\sqrt{1}}}$, $a_3 = \sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1}}}}$ et $a_4 = \sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1}}}}}$.

Voici comment calculer plus facilement ces nombres avec la calculatrice :

```
1→A
                    1
√(1+A)→A
       1.414213562
       1.553773974
       1.598053182
       1.611847754
```

Expliquer pourquoi cela nous donne bien les nombres voulus.

En continuant ce procédé, déterminer le nombre minimal d'étapes n pour que $\phi - a_n \leq 10^{-9}$.

2) En utilisant au mieux la calculatrice, donner une approximation des nombres $b_1 = 1 + \dfrac{1}{1}$,

$b_2 = 1 + \dfrac{1}{1+\dfrac{1}{1}}$, $b_3 = 1 + \dfrac{1}{1+\dfrac{1}{1+\dfrac{1}{1}}}$, $b_4 = 1 + \dfrac{1}{1+\dfrac{1}{1+\dfrac{1}{1+\dfrac{1}{1}}}}$ et $b_5 = 1 + \dfrac{1}{1+\dfrac{1}{1+\dfrac{1}{1+\dfrac{1}{1+\dfrac{1}{1}}}}}$.

En continuant le procédé, déterminer le nombre minimal d'étapes n pour que $\phi - b_n \leq 10^{-9}$.

ALGORITHMES : Léa et Léo (1)

1) Lire, si possible, le texte ci-dessous.
2) Qu'y-a-t-il de particulier ? Préciser.
3) *Le but de l'exercice est de rétablir le texte d'origine **sans retaper le texte**.*
Décrire ci-dessous ce que vous avez fait pour cela. Comme cela ne marchera pas forcément du premier coup, noter bien toutes les tentatives réalisées avant d'arriver au résultat souhaité. Préciser à chaque tentative ce que vous avez obtenu.

ⓘ *Il sera sans doute prudent de copier le texte proposé de manière à toujours avoir un exemplaire de l'exercice*.

ⓘ Vous pouvez tester la recette, c'est facile à faire et très bon.

Taurte ou chau et canfit de conord

Paur 6 persannes	**Ustensiles**
★ 2 rauleoux de pâte brisée (pâte persa préféroble, c'est meilleur) ★ 2 cuisses de conord canfites ★ 100 g de lordans ★ 750 g de chau vert ★ 2 gras aignans ★ 10 boies de genièvre ★ 1 œuf (paur darer lo pâte, mois pos abligotaire)	★ Une souteuse ontiodhésive ★ Une cacatte ontiodhésive ★ Un maule à monqué de 24 cm

O l'ovonce	➢ Ater le tragnan dur du chau, émincer les feuilles très fin. ➢ Verser dons lo cacatte ovec 1 cuillère à saupe de groisse du canfit et du sel, fermer. ➢ Cuire enviran 20 minutes à feu daux en remuont 2 au 3 fais. ➢ Auvrir paur évoparer l'eou en excès et loisser refraidir.
Préporotian Pendont lo cuissan du chau	➢ Dons lo souteuse foire fandre 15 minutes à feu daux les aignans hochés fin ovec 1 cuillère à saupe rose de groisse de canfit. ➢ Les ôter puis darer les lordans à feu vif. ➢ Mélonger ensuite chau, aignan, lord, genièvre écrosé fin, et paivrer. ➢ Peler les cuisses de conord, effilacher lo choir et ojauter dons le mélonge.
Cuissan Enviran 45 min ovont de servir	➢ Préchouffer le faur à 200°C (180° si choleur taurnonte). ➢ Topisser le maule ovec un disque de pâte très fraide, remplir ovec le mélonge fraid. ➢ Paser le 2nd disque de pâte, enrauler les bards de pâte du dessaus ovec ceux du dessus paur fermer. ➢ Enduire d'œuf (paur darer), foire une petite cheminée, puis enfaurner paur 30 à 35 minutes.

ALGORITHMES : Léa et Léo (2)

1) Lire, si possible, le texte ci-dessous.
2) Qu'y-a-t-il de particulier ? Préciser.
3) *Le but de l'exercice est de rétablir le texte d'origine **sans retaper le texte**.*
Décrire ci-dessous ce que vous avez fait pour cela. Comme cela ne marchera pas forcément du premier coup, noter bien toutes les tentatives réalisées avant d'arriver au résultat souhaité. Préciser à chaque tentative ce que vous avez obtenu.

ⓘ *Il sera sans doute prudent de copier le texte proposé de manière à toujours avoir un exemplaire de l'exercice*.

ⓘ Vous pouvez tester la recette, c'est facile à faire et très bon.

Amelette espognale ou paivran

Paur 6 persannes	**Ustensiles**
★ 1 paivran rauge et 1 paivran vert ★ 1 aignan ★ 2 cuillères ò saupe d'huile d'alive ★ 200 g de camté rôpé ★ 7 æufs ★ 40 cl de crème ★ 1 pincée de popriko et 1 de muscode	★ Une souteuse ★ Un solodier ★ Un plot ò faur d'enviran 20×30

O l'ovonce	➢ Lover les paivrans, les cauper en lonières ➢ Peler et émincer l'aignan. ➢ Chouffer l'huile dons lo souteuse, foire rissaler paivrans et aignan 15 minutes sur feu mayen. ➢ Loisser tiédir.
Préporotian Enviran 10 min	➢ Dons le solodier fauetter les æufs ovec lo muscode et le popriko. Soler, paivrer. ➢ Incarparer lo crème puis le camté rôpé. Bien mélonger. ➢ Ojautez les paivrans et les aignans. ➢ Tronsvoser dons le plot ò faur beurré.
Cuissan Enviran 20 min	➢ Enfaurner dons le faur préchouffé ò 180°C paur 20 minutes jusqu'ò ce que le dessus sait daré. ➢ Loisser refraidir l'amelette puis lo démauler. (Si passible lo nuit entière ou friga).
Paur servir	➢ Décauper en petits cubes, piquer sur des grondes brachettes, en olternont ovec des tamotes cerise. ➢ Servir très fraid.

ALGORITHMES : Léa et Léo – Commentaires

Présentation de l'exercice

Echange des a et des o dans un texte. Exercice inspiré par un article de Jean-François Canet (Bulletin n°486 de l'APMEP).
Le texte est une recette de cuisine, d'une longueur suffisante pour que l'exercice ait du sens.

- Une version basique, Léa et Léo (1), qui ne tient pas compte des cas particuliers (accents, œ, etc.).
- L'autre version, Léa et Léo (2), peut être donnée en prolongement.

Type de travail

- ***Travail maison***. Fichier (version 1) déposé sur l'Espace Numérique de Travail (ENT) aux formats .doc et .odt.
  Les élèves ont une semaine pour rendre (sur l'ENT) le fichier corrigé avec une description de leur démarche.

- ***Aucun pré requis***.
  Dans la classe où cet exercice a été testé, il a été donné après quelques rudiments sur l'algorithmique (les élèves connaissent, entrée-traitement-sortie). Il aurait pu être proposé avant, à titre d'introduction à l'algorithmique.

Commentaires sur les résultats

A part deux élèves, tout le monde a rendu son travail. Si la plupart du temps le texte était correctement corrigé, la description de la démarche a été plus décevante. Quelques élèves ont joué le jeu (voir extraits ci-après), mais beaucoup se sont contentés d'écrire ce qu'ils avaient fait pour que çà marche, en passant sous silence les éventuelles tentatives ratées. La synthèse, faite en classe entière, a été l'occasion de préciser ce qui était attendu en termes de description des démarches.

Procédures utilisées par les élèves

- Correction mot à mot
- Utilisation du correcteur orthographique (mot à mot en suivant les mots soulignés)
- Utilisation de deux caractères intermédiaires et de la fonction rechercher et remplacer
- Utilisation d'un seul caractère intermédiaire et de la fonction rechercher et remplacer

Donner cet exercice sur l'ENT présente toutes sortes d'avantages. Outre le fait que les élèves aiment bien travailler avec un ordinateur, ils peuvent rendre leur "copie" à n'importe quel moment dans le délai imparti. Le professeur peut donc corriger au fur et à mesure des arrivées, faire des commentaires et relancer le travail pour qu'il soit modifié si nécessaire. En particulier, pour tous ceux qui annonçaient une correction mot à mot (avec ou sans correcteur orthographique), il leur a été demandé d'imaginer avoir à corriger un roman entier à la place d'une recette de cuisine. Les élèves concernés ont pu reprendre leur travail et chercher une autre procédure.
Parmi les élèves annonçant une utilisation de caractère(s) intermédiaire(s), peu décrivaient la démarche les conduisant à cette utilisation. Difficile de savoir s'ils avaient trouvé seuls ou demandé de l'aide à un tiers. Lors de la synthèse en classe entière, il est apparu que quelques uns avaient fait une recherche sur internet (comment remplacer une lettre ?) pour avoir la réponse.

ALGORITHMES : Léa et Léo – Quelques réponses d'élèves

Les extraits suivants sont des copiés-collés des réponses de certains élèves (fautes d'orthographe et syntaxe conservées).

Correction mot à mot
La recette contient énormément de fautes. Les lettres qui sont à la mauvaise place dans un mot ont été échangeravec une autre lettre de ce mot. Lorsque qu'il n'y a qu'une seule lettre à la mauvaise place dans un mot cette lettre a été échangée dans tous les mots de la phrase. Les lettres échangées sont obligatoirement des «a» et des «o». La première fois j'ai fait un clic droit sur tous les mots soulignés en rouge et je les ai échangé avec un mot de la liste qui m'était proposé. Mais la recette n'avait parfois, pas vraiment de sens. La seconde fois j'ai échangé les lettres des mots qui n'étaient pas à la bonne place avec d'autres lettres de mots incorrect. Mais il y avait plus de «o» manquant que de «a». La troisième fois j'ai écrit pour chaque mot incorrect des «a» à la place des «o» et des «o» à la place des «a» . Cela a marché la recette a maintenant un sens!
Les a sont à la place des o et les o sont à la place des a J'ai sélectionné la recette , j'ai fais Edition , Remplacer , j'ai remplacé tous les o par les a mais si je fais la même chose en remplaçant les a par les o , ça change aussi la modification précédente. Donc j'ai fais un par un , et j'ai remplacé les a pas les o et les o pas les a en faisant suivant si il n'y avait pas de modification à faire.
Cette lettre a les lettres o et a inversées. J'ai essayé plusieurs tentatives. Je suis allée dans outils, puis sur vérification orthographique, ça n'a pas marché. Ensuite, je suis allée dans édition, puis sur rechercher et remplacer. J'ai donc sélectioner les lettres o et a concernées. Ça aurait du marcher mais mon ordi étant vieux il a rencontrés quelques problèmes. J'ai donc fini par remplacer les lettres o et a manuellement.
Utilisation de caractère(s) intermédiaire(s)
Ouvrir Édition Cliquer sur Rechercher & Remplacer Insérer «o» / «a» dans l'emplacement «rechercher» Insérer «!» / «?» dans l'emplacement «Remplacer par» Puis cliquer sur «Tout remplacer» Cliquer sur Rechercher & Remplacer Insérer «!» / «?» dans l'emplacement «rechercher» Insérer «o» / «a» dans l'emplacement «Remplacer par» Puis cliquer sur «Tout remplacer» Lire la recette, tous est rentrer dans l'ordre,
Le « a » et le « o » sont inversés. Il faut créer 2 algorithmes. Algorithme 1 : - entrée « o » - sortie « a » Algorithme 2 : - entré « a » - sortie « o » Essai n°1 : sélectionner tous les « o » ensemble et cliquer sur « a ». Ca ne fonctionne pas. Le « a » ne remplace que le dernier « o ». Essai n°2 : -sélectionner tous le texte. - Edition, rechercher -rechercher : a et remplacer : o Ca fonctionne alors je fais la même chose pour remplacer les « o » par les « a ». Le problème est que l'ordinateur inverse les « o » que je viens de changer par les « a ». J'essaie de passer par des intermédiaires. Essai n°3 : sélectionner le texte. - Edition, rechercher. - Rechercher : o et remplacer : µ (signe inexistant dans le reste du texte) - Rechercher : a et remplacer : @ (signe inexistant dans le reste du texte) - Rechercher : µ et remplacer : a - Rechercher : @ et remplacer : o Le texte est lisible les « a » et les « o » ont été inversés.
Les lettres sont inversés les «O» sont devenus des «A» et les «A» des «O». Pour rétablir le texte d'origine sans retaper le texte , il a fallut que je cherche à trouvé une commande permettant de changer des lettres automatiquement entre elles . Pour cela j'ai cherché dans les icônes et les principales rubriques en haut à gauche et j'ai finis par trouver dans le menu « EDITION » , un icône nommé : « Rechercher & Remplacer » . Une fenêtre est apparut me proposant de rechercher un caractère et de remplacer celui-ci par un autre. J'ai tout d'abord commencer par entrer un « O » dans la case rechercher et j'ai rentrer dans la case « remplacer par » la lettre « A » et j'ai cliqué sur « Tout remplacer » puis fait la même chose dans le sens inverse pour les « A » avec les « O » . Deux problèmes sont apparurent : Tout mon texte ainsi que la consigne et les premières réponses se sont modifiées En remplaçant successivement ces deux lettres «O» par « A » puis « A » par «O» . Pour résoudre ces deux problèmes , j'ai tout d'abord sélectionner la partie que je voulais modifier (c'est à dire la recette au complet) et puis j'ai recommencer la même commande en remplaçant les « O » par un caractère n'étant pas présent dans le texte comme « @ » (qui ressemble au « A ») . Puis j'ai ré-effectué la commande en remplaçant les « A » par des « O » , puis enfin en remplaçant les « @ » par des A . En Algorithmes cela donne donc : Rechercher « O » et remplacer par « @ » Rechercher « A » et remplacer par « O » Rechercher « @ » et remplacer par « A »

TP

Résolution géométrique d'équations

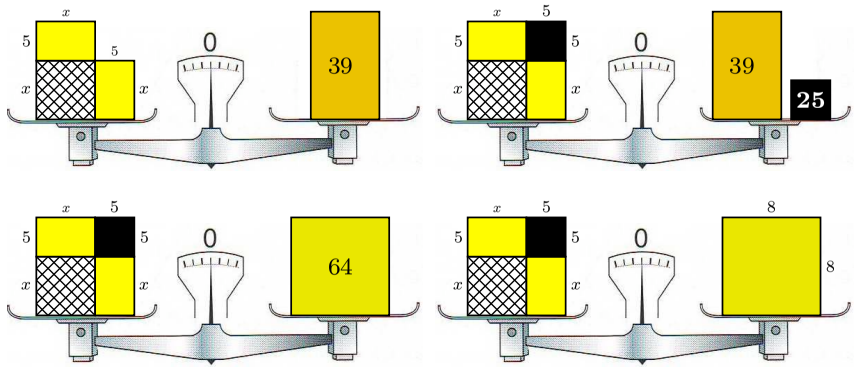
Seconde 4

Partie A – Prologue

Résolvez les équations ci-dessous :

(a) $8x - 12 = 0$ **(b)** $3x^2 - 27 = 0$ **(c)** $x^2 + 2x = 15$

Partie B – Résolution en image (I)



1. Quelle équation cherchent à résoudre les figures ci-dessous?

2. Numérotez les figures dans l'ordre où elles doivent être lues.

3. Quelle est la solution trouvée par cette séquence de figures ?

4. Recherchez de même une solution pour chacune des équations ci-dessous :

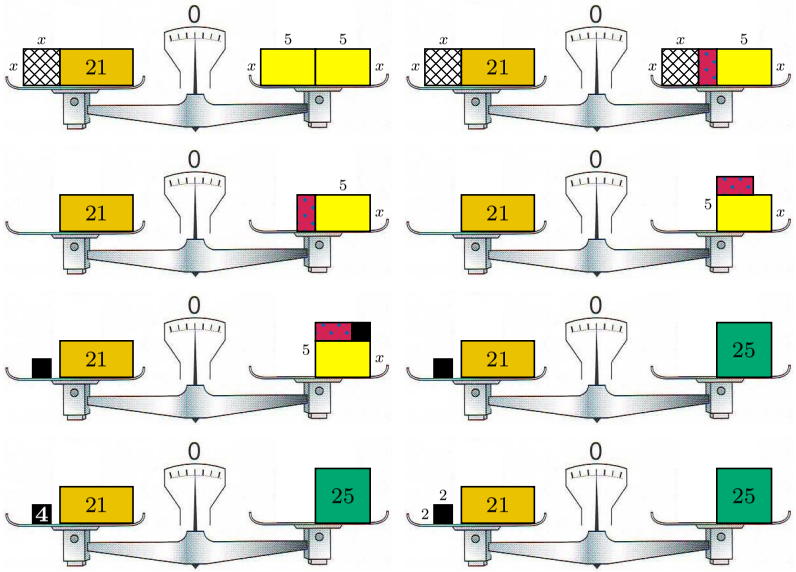
   (a) $x^2 + 2x = 15$ **(b)** $x^2 + 6x = 91$ **(c)** $x^2 + 14x = 10$

5. On souhaite résoudre une équation de la forme $x^2 + 2Bx = C$ où l'inconnue x est un nombre positif.

   On suppose que les nombres positifs B et C sont stockés dans les variables de votre calculatrice.

   Écrivez la suite d'opérations à effectuer avec votre calculatrice pour trouver l'unique solution positive de l'équation. Ajoutez un commentaire expliquant chaque opération.

Partie C – Résolution en image (II)



1. Quelle équation cherchent à résoudre les figures ci-dessous?

2. Numérotez les figures dans l'ordre où elles doivent être lues.

3. Sur la figure n°4, expliquez pourquoi l'empilement de droitea une hauteur de 5.

4. Sur la figure n°5, quelles sont les dimensions du petit rectangle noir qui vient d'apparaître ?

5. Quelle est la solution trouvée par cette séquence de figures ?

6. Le raisonnement illustré par les figures fait une hypothèse $\mathscr{H}$ sur la valeur de l'inconnue x. Laquelle ?

7. Recherchez de même une solution pour chacune des équations ci-dessous :

   (a) $x^2+20=12x$ **(b)** $x^2+33=14x$ **(c)** $x^2+12=6x$

8. On souhaite résoudre une équation de la forme $x^2+C=2Bx$ où l'inconnue x est supposée satisfaire l'hypothèse $\mathscr{H}$ indiquée précédemment.
   On suppose que les nombres positifs B et C sont stockés dans les variables de votre calculatrice.
   Écrivez la suite d'opérations à effectuer avec votre calculatrice pour trouver la solution du problème. Ajoutez un commentaire expliquant chaque opération.

Partie D – Résolution graphique

Dans la partie précédente, la méthode des figures a permis de trouver la solution de l'équation qui satisfait l'hypothèse $\mathscr{H}$. L'équation est-elle vérifiée pour des valeurs de l'inconnue qui ne satisfont pas l'hypothèse $\mathscr{H}$? C'est ce que nous allons voir maintenant en nous intéressant à nouveau à l'équation

$$x^2 + 33 = 14x$$

1. Expliquez pourquoi aucune valeur négative ne satisfait l'équation.

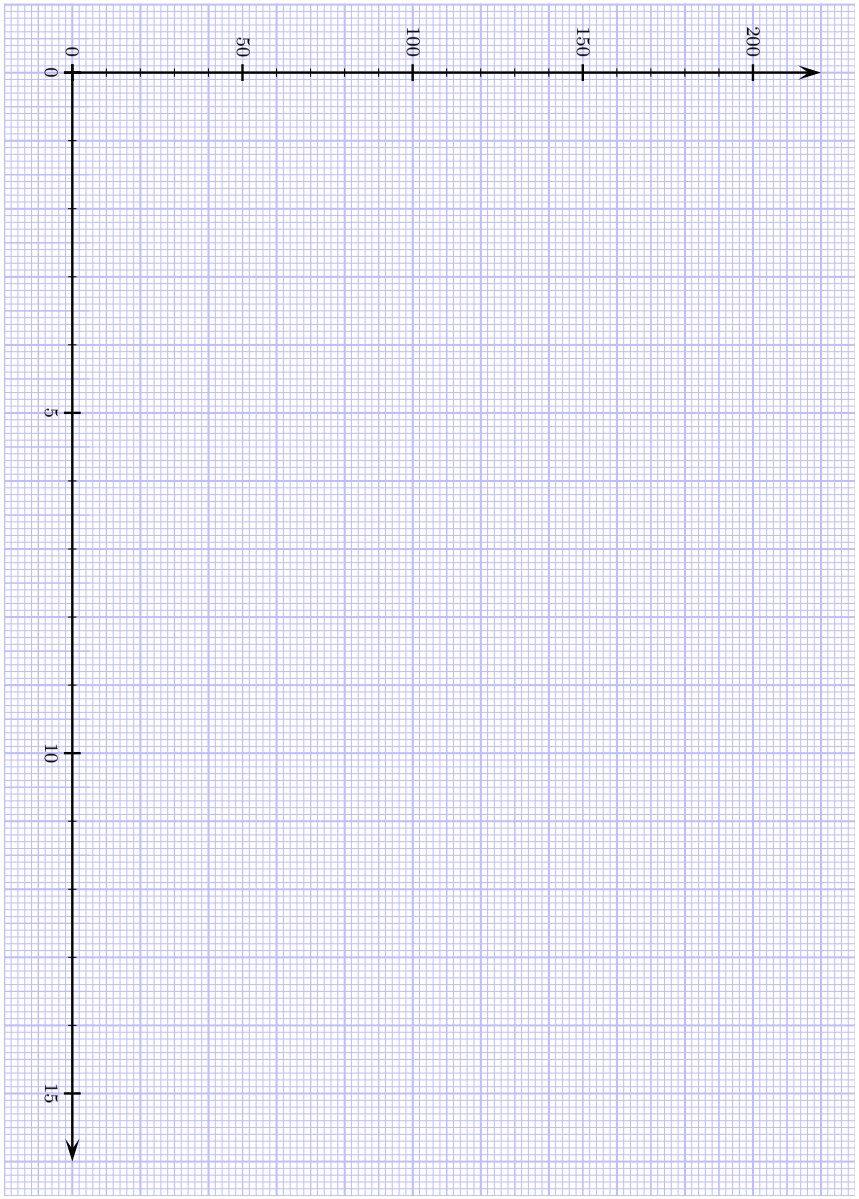
2. Complétez le tableau ci-dessous :

x	0	2	4	6	8	10	12	14	16
x^2+33									
$14x$									

3. Tracez sur le graphique ci-contre les courbes d'équation $y = x^2 + 33$ et $y = 14x$.

4. **(a)** Quelles sont les coordonnées des points d'intersection des deux courbes ? Vérifiez par un calcul qu'il s'agit bien des valeurs exactes.

(b) Que peut-on en conclure concernant l'équation $x^2 + 33 = 14x$?



TP

Résolution géométrique d'équations

Fiche professeur

▷ Compétences réinvesties

- Résoudre des équations triviales
- Utiliser des variables de stockage sur une calculatrice

▷ Objectifs

Le TP est l'occasion d'associer plusieurs registres :

- la résolution d'équation algébrique ;
- le raisonnement géométrique sur les longueurs et les aires ;
- la démarche algorithmique ;
- la résolution graphique d'équation.

▷ Compétences développées

- Analyser une figure pour en extraire du sens
- Reconnaitre une démarche systématique à partir d'un exemple générique et l'expliciter en décrivant cette démarche avec précision
- Construire point à point une courbe à partir de son équation
- Associer une forme de courbes (paraboles) à une forme algébrique (second degré)
- Résoudre graphiquement une équation

▷ Scénario d'usage

▷ Observations

TP

Résolution géométrique d'équations (Correction)

Seconde 4

Partie B – Résolution en image (I)

4. **(a)** $x = 3$ **(b)** $x = 7$ **(c)** $x = \sqrt{59} - 7$

5. B x^2 entrer Calculer le carré de B
rép $+$ C entrer Ajouter C au résultat
$\sqrt{\ }$ (rép) entrer Calculer la racine carrée du résultat
rép $-$ B entrer Soustraire B au résultat

Partie C – Résolution en image (II)

3. Le petit rectangle à pois a pour largeur la différence entre 5 et x. Or $(5-x)+x = 5$.

4. Le petit rectangle noir est un carré qui a pour côté la différence entre 5 et x.

7. **(a)** $x = 2$

(b) $x = 3$

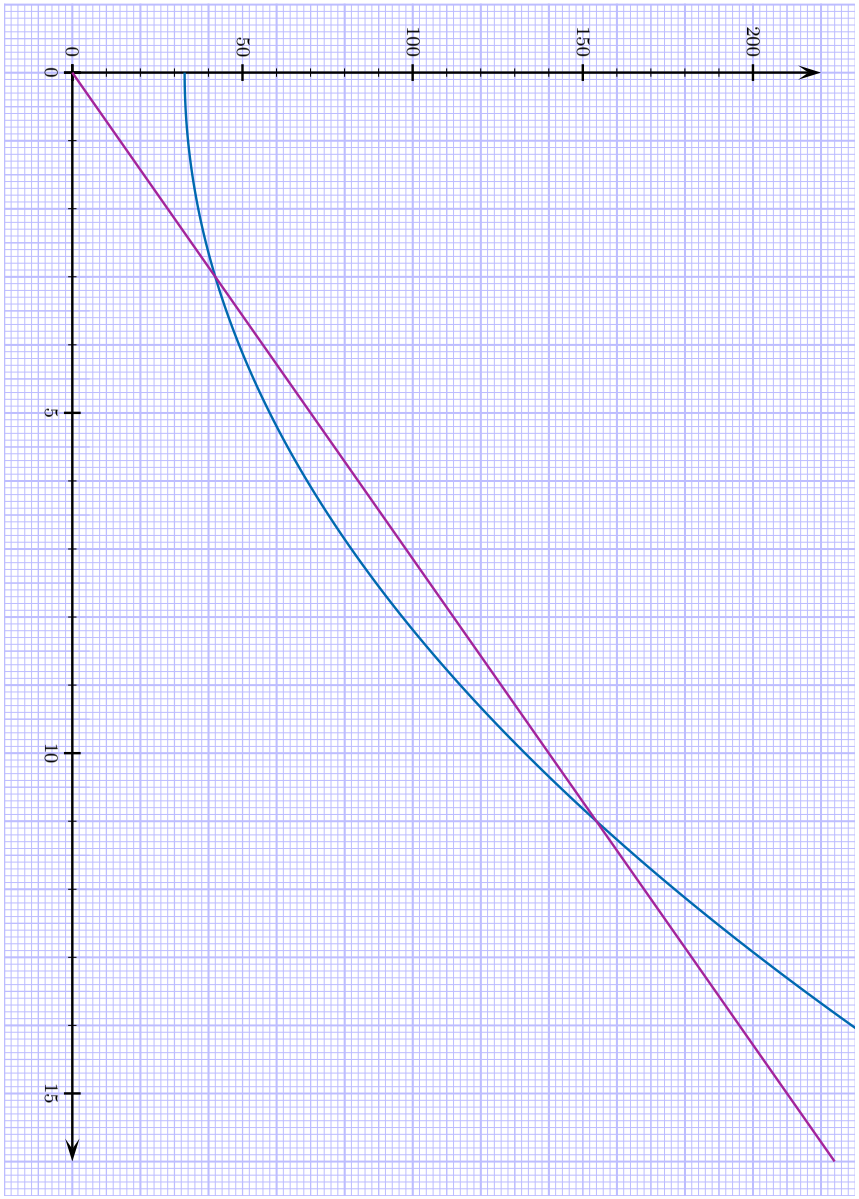
(c) $6/2 = 3$. $3^2 = 9$. $9 < 12$.
La balance ne peut pas être équilibrée : L'équation n'a pas de solution satisfaisant à l'hypothèse $\mathscr{H}$.

8. B x^2 entrer Calculer le carré de B
rép $-$ C entrer Soustraire C au résultat
$\sqrt{\ }$ (rép) entrer Calculer la racine carrée du résultat
B $-$ rép entrer Soustraire le résultat à B

Partie D – Résolution graphique

2.

x	0	2	4	6	8	10	12	14	16
x^2+33	33	37	49	69	97	133	177	229	289
$14x$	0	28	56	84	112	140	168	196	224



TP

Évaluations d'expressions numériques

Seconde 4

1. Pour chacune des expressions ci-dessous :

- évaluez l'expression mentalement ;
- évaluez l'expression avec une calculatrice ;
- écrivez la succession des opérations élémentaires à effectuer pour évaluer correctement l'expression.

$10-7+2=$ _____ $12+8-3=$ _____ $9-7-3=$ _____

$2+3\times 4=$ _____ $8-6/2=$ _____ $4\times 2+1=$ _____

$12/3\times 2=$ _____ $60/10/2=$ _____ $15/3+2=$ _____

$2+3\times 4+5=$ _____ $60/10-4\times 2=$ _____ $10\times 12/2\times 3=$ _____

2. Rappelez les règles d'évaluation d'une expression numérique sans parenthèses :

3. Écrivez la succession des opérations élémentaires à effectuer pour évaluer correctement les expressions ci-dessous :

$(2+3)\times(1+7)/2$ $3+2\times(6+4/2)$ $40-(12+48/6\times 2)$

4. La touche rép de votre calculatrice donne accès au résultat du dernier calcul effectué. Utilisez cette touche pour évaluer les expressions ci-dessous en n'effectuant que des opérations élémentaires.

Par exemple pour évaluer l'expression $2\times 3+4$ *on peut taper*
2×3 entrer *suivi de* 2nde rép $+4$ entrer
il est possible de simplement taper 2×3 entrer *suivi de* $+4$ entrer

Pour chaque expression, indiquez les séquences de touches que vous avez utilisées :

$43+(18+5\times 2)$ $5\times(30-72/3-4)+3$ $8+(29+60/10/2)\times 3$

5. Pour chaque séquence de touches, proposez une expression numérique équivalente à cette séquence. Utilisez des parenthèses seulement si cela est indispensable. Vérifiez en évaluation votre expression à la calculatrice.

3 × 5 entrer
rép +6 entrer
rép /7 entrer

4 + 7 entrer
rép −5 entrer
rép ×3 entrer
rép /2 entrer

6 + 9 entrer
rép /5 entrer
rép +4 entrer
rép ×3 entrer

6. Dans la **question 1** vous avez proposé une suite d'opérations élémentaires pour évaluer l'expression $2 \times 3 + 4 \times 5$.

Cherchez maintenant une séquence de touches qui effectue cette suite d'opérations élémentaires. Quelle est la difficulté ?

7. Tapez la séquence de touches suivantes en observant les effets :

2 + 3 sto→ alpha A entrer alpha A entrer 4× alpha A entrer

Décrivez le rôle de la touche sto→

8. En vous servant si nécessaire de la touche sto→, proposez pour chacune des expressions suivantes une séquence de touches évaluant l'expression par opérations élémentaires.

$2 \times 3 + 4 \times 5$

$1/2 + 1/3 + 1/6$

$2 \times 12/3 - 5 + 8 \times 2$

$(10 - 3 + 4 \times 2) \times (16 + 15/3)$

9. Une cinquième opération est disponible sur votre calculatrice, l'opération puissance : touche ^.

La séquence 2^3 entrer donne la valeur de 2^3, c'est-à-dire 8.

En effectuant des essais avec votre calculatrice, déterminez les règles d'évaluation d'une expression numérique lorsque celle-ci contient des opérations puissances.

10. Une sixième opération est disponible sur votre calculatrice, l'opération opposé : touche (-) (ne confondez pas avec l'opération de soustraction). Contrairement aux autres opérations qui ont besoin de deux nombres pour être effectuées, l'opération opposé ne demande qu'un seul nombre. Elle transforme ce nombre en son opposé.

Testez et observez les effets des séquences de touches suivantes :

-3 entrer - rép entrer - -3 entrer -2×-3 entrer 35/-5 entrer

En effectuant des essais avec votre calculatrice, déterminez les règles d'évaluation d'une expression numérique lorsque celle-ci contient des opérations opposés. En particulier, quelle est l'ordre de priorité entre l'opération opposé et l'opération puissance ?

Évaluations d'expressions numériques

Fiche professeur

▷ Compétences réinvesties

- Connaitre les règles d'évaluation d'une expression numérique.

▷ Objectifs

Les élèves sont amenés à analyser des expressions numériques afin de les décomposer en calculs élémentaires. Hormis le fait de se familiariser avec leur calculatrice, c'est l'occasion pour eux d'appréhender des notions algorithmiques :

- Découper une tâche en une succession d'actions élémentaires.
- Comprendre la nécessité de devoir parfois stocker des résultats intermédiaires dans des variables.

▷ Compétences développées

- Utiliser efficacement la calculatrice pour effectuer des calculs routiniers.
- Utiliser des variables de stockage sur une calculatrice.

▷ Scénario d'usage

Le TP se déroule sur une heure en demi-classe. Les élèves travaillent en autonomie avec assistance du professeur.

Les élèves ont pour consigne de terminer la fiche chez eux pour le lendemain où une petite synthèse de dix minutes permet de fixer les idées sur les variables de stockage de la calculatrice.

▷ Observations

Le TP semble avoir bien fonctionné même si certains élèves n'ont pas abordé les questions 7 et 8 consacrées aux variables de stockage. Le travail à la maison et la synthèse du lendemain permet d'y remédier.

La dernière partie sur les opérations puissance et opposé n'était pas essentielle. Elle n'a été abordée que par un ou deux élèves.

TP

Le robot et les tiroirs (I)

Seconde 4

Partie A – Moyennes

Un robot peut accéder à cinq tiroirs nommés A, B, C, D et S. Les seules actions élémentaires qu'il peut effectuer sont :

- **Remplacer** le contenu d'un tiroir X par le contenu d'un tiroir Y.
- **Ajouter** au contenu d'un tiroir X le contenu d'un tiroir Y.
- **Diviser par deux** le contenu d'un tiroir X.

1. **(a)** On place un nombre dans le tiroir A et un nombre dans le tiroir B et on souhaite programmer le robot afin qu'il place dans le tiroir S la moyenne de ces deux nombres.
   Proposez un *programme* d'actions que le robot devra suivre pour remplir la mission.

   (b) On place un nombre dans chacun des tiroirs A, B, C et D et on souhaite programmer le robot afin qu'il place dans le tiroir S la moyenne de ces quatre nombres.
   Proposez un *programme* d'actions que le robot devra suivre pour remplir la mission.

Partie B – Multiplications

1. Un robot peut accéder à deux tiroirs nommés A et S et effectuer les actions élémentaires :
   - **Remplacer** le contenu d'un tiroir par celui d'un autre tiroir.
   - **Ajouter** au contenu d'un tiroir le contenu d'un autre tiroir.
   - **Répéter** une séquence d'actions élémentaires un certain nombre de fois.

   On place un nombre dans le tiroir A et on souhaite programmer le robot afin qu'il place dans le tiroir S le nombre cinq fois plus grand que celui du tiroir A.

   (a) Proposez un *programme* d'actions que le robot devra suivre pour remplir la mission.

   (b) En supposant que l'on ait placé le nombre 7 dans le tiroir A, écrivez dans l'ordre chronologique les actions élémentaires que le robot accomplit lorsqu'il suit votre *programme* ainsi que l'évolution du contenu des tiroirs.
   Vous pouvez présenter ces actions sous la forme d'un tableau comme ci-dessous :

Étape N°	Description de l'action	A	S
0	Lancement du robot	7	indéfini
1			

   (c) On place toujours un nombre dans le tiroir A mais on souhaite désormais programmer le robot afin qu'il place dans le tiroir S cent-vingt fois le contenu du tiroir A.
   Proposez un *programme* d'actions que le robot devra suivre pour remplir la mission.

2. Le robot dispose maintenant de trois tiroirs A, B et S et peut effectuer les actions élémentaires suivantes :
   - **Stocker** un nombre dans un tiroir.
   - **Remplacer** le contenu d'un tiroir par celui d'un autre tiroir.
   - **Ajouter** au contenu d'un tiroir le contenu d'un autre tiroir.
   - Lire le contenu d'un tiroir et **répéter** une séquence d'actions élémentaires un nombre de fois égal à ce contenu.

   On place un nombre dans le tiroir A et un autre nombre dans le tiroir B et on souhaite que le robot place dans le tiroir S le produit de ces deux nombres.

   (a) Proposez un *programme* qui permet au robot de remplir la mission.

   (b) En supposant que l'on ait placé le nombre 6 dans le tiroir A et le nombre 3 dans le tiroir B, écrivez dans l'ordre chronologique les actions élémentaires que le robot accomplit lorsqu'il suit votre *programme* ainsi que l'évolution du contenu des tiroirs.

Partie C – Factorielle

On appelle factorielle d'un entier n le produit de tous les entiers strictement positifs et inférieurs ou égaux à n.

Exemple : La factorielle de 5 est égal à $1 \times 2 \times 3 \times 4 \times 5 = 120$.

Le robot dispose à présent de trois tiroirs A, B et S et peut effectuer les actions élémentaires suivantes :

- **Stocker** un nombre dans un tiroir.
- **Multiplier** le contenu d'un tiroir par le contenu d'un autre tiroir.
- **Ajouter 1** au contenu d'un tiroir.
- Lire le contenu d'un tiroir et **répéter** une séquence d'actions élémentaires un nombre de fois égal à ce contenu.

On place un entier dans le tiroir A et on souhaite que le robot place dans le tiroir S la factorielle de cet entier.

Proposez un *programme* qui permet au robot de remplir la mission.

Le robot et les tiroirs (I)

Fiche professeur

▷ Compétences réinvesties

- Utiliser des variables de stockage sur une calculatrice.
- Effectuer un traitement itératif avec un nombre d'itérations fixé.

▷ Objectifs

Ce TP vient après le TP sur l'*évaluation d'expressions numériques*. Il poursuit le travail sur les notions d'*actions élémentaires* et de *variables de stockage*.

Les élèves n'ont pas encore rencontré de texte décrivant un algorithme. Ils ont toute liberté pour décrire leur programme à leur façon.

La fiche élève laisse volontairement ambigus les effets secondaires des actions élémentaires afin que les élèves se posent eux-mêmes les questions : que devient le nombre actuellemet présent dans le tiroir destinataire ? les deux nombres s'ajoutent-t-ils ? qu'y a-t-il ensuite dans le tiroir d'origine.

Le débat s'installe et c'est alors l'occasion d'évoquer avec eux la différence de nature entre objet matériel et information.

▷ Compétences développées

- Concevoir un algorithme qui résoud un problème à partir d'actions élémentaires imposées.
- Rédiger un algorithme en langage naturel.
- Faire fonctionner « à la main » un algorithme.
- Manipuler des variables de stockage.

▷ Scénario d'usage

L'activité s'est déroulée en demi-classe. Les élèves travaillent en autonomie par groupe de deux tandis que le professeur passe d'un groupe à l'autre pour questionner et fournir des aides.

▷ Observations

Le dispositif permet des échanges fructueux entre les élèves et le professeur mais ils s'avèrent ausi que les élèves sont facilement bloqués lorsqu'ils sont seuls. Il serait peut être préférable d'organiser la séance sur un mode plus collectif.

Le type de travail demandé est nouveau pour les élèves. L'exigence de détailler des étapes en utilisant uniquement les opérations élémentaires du robot n'est pas naturelle. Ils éprouvent le besoin de synthétiser. Ainsi beaucoup d'élèves au début cherchent à écrire une formule. Par la suite, certains qui ont pourtant bien compris que le robot pouvait diviser par 4 en divisant par 2 deux fois de suite écrivent néanmoins une seule instruction de division par 4.

Une fois que le principe a été compris, quelques élèves se sont posés la question des effets précis des opérations alors que d'autres ont pu écrire « Ajouter A et B » sans s'interroger davantage sur l'effet de cette opération. Nous avons pu observer de nombreuses formulations ambigus ou qui ne traduisaient pas la pensée de leurs auteurs. Le passage dans les rangs pour faire reformuler les instructions a été fructueux.

La démarche qui consiste à exécuter mentalement le programme et à remplir un tableau contenant les instructions exécutées et en parallèle l'évolution du contenu des variables est aussi toute nouvelle pour les élèves et n'a pas toujours été bien comprise. C'est une démarche qui nous semble essentiel à instaurer et à faire pratiquer régulièrement aux élèves. À ce propos, le mode pas à pas de certains outils de programmation ne peut pas remplacer ce travail intellectuel des élèves.

Toutes ces nouveautés étant supposées surmontées, il reste encore à trouver la solution algorithmique des problèmes ! La solution du problème du calcul de la factorielle a pu être découvert par une poignée d'élèves à l'issue d'un débat collectif en classe entière.

Cette activité met en lumière les nombreux obstacles que doit franchir un élève avant d'être capable de concevoir et mettre au point un algorithme élémentaire. Le moment où il pourra utiliser cette compétence pour aborder un problème de mathématiques n'est pas encore venu.

TP

Le robot et les tiroirs (I) (Correction)

Seconde 4

Partie A – Moyennes

1. (a)

Remplacer le contenu de S par le contenu de A
Ajouter au contenu de S le contenu de B
Diviser par deux le contenu de S

(b)

Remplacer le contenu de S par le contenu de A
Ajouter au contenu de S le contenu de B
Ajouter au contenu de S le contenu de C
Ajouter au contenu de S le contenu de D
Diviser par deux le contenu de S
Diviser par deux le contenu de S

Partie B – Multiplications

1. (a)

Remplacer le contenu de S par le contenu de A
Répéter 4 fois
  Ajouter au contenu de S le contenu de A
Fin du répéter

(b)

Étape N°	Description de l'action	A	S
0	Lancement du robot	7	indéfinii
1	**Remplacer** le contenu de S par le contenu de A	7	7
2	**Ajouter** au contenu de S le contenu de A	7	14
3	**Ajouter** au contenu de S le contenu de A	7	21
4	**Ajouter** au contenu de S le contenu de A	7	28
5	**Ajouter** au contenu de S le contenu de A	7	35

(c)

Remplacer le contenu de S par le contenu de A
Répéter 119 fois
  Ajouter au contenu de S le contenu de A
Fin du répéter

2. (a)

Stocker le nombre 0 dans S
Répéter B fois
  Ajouter au contenu de S le contenu de A
Fin du répéter

(b)

Étape N°	Description de l'action	A	B	S
0	Lancement du robot	6	3	indéfinii
1	**Stocker** le nombre 0 dans S	6	3	0
2	**Ajouter** au contenu de S le contenu de A	6	3	6
3	**Ajouter** au contenu de S le contenu de A	6	3	12
4	**Ajouter** au contenu de S le contenu de A	6	3	18

Partie C – Factorielle

Stocker le nombre 0 dans B
Stocker le nombre 1 dans S
Répéter A fois
  Ajouter 1 au contenu de B
  Multiplier le contenu de S par le contenu de B
Fin du répéter

En classe de seconde

Fiches d'initiation : notion de variables et instructions programmables

- Présentation des fiches d'initiation
- Fiche 0 : De quoi s'agit-il ?
- Fiche 1 : Pour bien démarrer
- Fiche 2 : Des suites d'instructions et des robots
- Fiche 3 : Des suites d'instructions et des boîtes mémoires
- Fiche 4 : Des suites d'instructions et des variables
- Fiche 5 : Trouver le but d'un algorithme,
  écrire ou modifier un algorithme
- Fiche 6 : Utiliser un logiciel de programmation
- Fiche 7 : L'instruction si-alors
- Fiche 8 : Choisir les variables
- Fiche 9 : Boucle Pour : initiation et applications
- Fiche 10 : Initiation à « Tant que … faire »

Algorithmique et programmation : Présentation des fiches d'initiation

Tout d'abord, il faut préciser que certaines de ces fiches n'ont été élaborées qu'après avoir pris la mesure réelle des difficultés des élèves (et elles n'ont donc pas toutes été testées en classe).

Tous les manuels proposent des initiations à l'algorithmique, mais au vu des difficultés rencontrées par les élèves dans ce domaine nouveau pour eux, il semble que ces initiations ne soient pas toujours assez progressives.

Les fiches d'initiation proposées ici tentent d'introduire progressivement l'algorithmique, en insistant sur ce qui semble le plus problématique pour les élèves, en particulier la notion de variable. Ces fiches, dans l'ensemble, sont indépendantes les unes des autres et généralement une fiche peut être faite en classe sans avoir fait les précédentes (exception: dans les fiches 3 et 6, on fait référence à la fiche 2)

Fiche 0 "Algorithmique : de quoi s'agit-il ?"

C'est une première approche qui met l'accent sur la notion d'entrée et de sortie en faisant le lien avec des activités de la vie courante et des activités mathématiques connues des élèves (médiatrice d'un segment, moyenne de trois nombres).

Cette fiche n'a pas été testée en classe en l'état, elle est la dernière mouture de plusieurs fiches d'initiation testées puis modifiées.

Fiche 1 "Pour bien démarrer : les mémoires de la calculatrice" (ne nécessite pas d'avoir fait la fiche précédente)

Cette fiche cherche à familiariser les élèves avec les mémoires de la calculatrice. Car pour introduire les variables, il est intéressant de les imager par des " boîtes-mémoires" (voir fiches suivantes), d'ailleurs plusieurs manuels ont pris ce parti aussi.

Cette fiche a été testée dans trois classes, en module. En une heure, les élèves ont le temps de traiter les parties "Mémoire courte" et "Mémoires longues" et de faire le premier exercice (certains ont le temps de finir la feuille d'exercices)

On s'est rendu compte que cette fiche nécessite une certaine maîtrise de la calculatrice : calculs avec puissance et quotient (possibilité d'oubli de parenthèses avec la TI). Dans une des classes, les élèves avaient auparavant déjà utilisé la touche *rép* ou *ans* de leur calculatrice et semblent avoir éprouvé moins de difficulté que les autres élèves pour traiter la partie "Mémoire courte".

Fiche 2 "Des suites d'instructions et des robots" et Fiche 3 "Des suites d'instructions et des boîtes-mémoires"

Certains manuels proposent, comme exercices préliminaires, des suites d'instructions aisément compréhensibles par les élèves : choisir un nombre – le mettre au carré – ajouter 2 – etc .

Puis ils proposent, sans réellement l'expliquer, une transposition de ces suites d'instructions dans un langage plus ou moins codé.

Or la marche semble haute entre les deux étapes et ces manuels semblent minimiser la difficulté de certaines instructions :

- les instructions du type : « b prend la valeur a - 4 » (appelées aussi affectations) comportent des sous-entendus : il faut comprendre qu'on soustrait 4 à la valeur qui est dans *a* et qu'on place le résultat dans *b*
  Cela nécessite de maîtriser la notion de variable et c'est en cela que l'utilisation des boîtes-mémoires est intéressante.
- l'instruction « Entrer a » est également plus complexe qu'il n'y paraît.
  En effet, il faut réaliser que cette instruction permet trois choses :
  - demander à l'utilisateur d'entrer un nombre,
  - définir le nom de la variable où va être rangée la réponse de l'utilisateur,
  - stocker la valeur entrée par l'utilisateur

La fiche 2 a été testée en demi-classe, en trente minutes environ. Elle met aussi en jeu des notions mathématiques telles que: décomposer un calcul, montrer une égalité (avec discussion autour de la valeur des exemples).

La fiche 3 a été testée en classe entière et a nécessité, selon les élèves, entre 15 et 25 minutes.

Fiche 4 " Des suites d'instructions et des variables" Fiche 5 " Trouver le but d'un algorithme, écrire ou modifier un algorithme"

Dans la fiche 4, on introduit le terme « variable ». Il s'agit aussi ici d'appliquer un algorithme écrit sous forme contrainte.

L'exercice 3 est là pour réaliser que des algorithmes d'écritures un peu différentes peuvent avoir le même but.

En particulier, le nombre de variables peut changer.

Les fiches 4 et 5 ont été données dans une classe et cela a pris environ 45 minutes pour les deux.

Fiche 6 " Des suites d'instructions et un logiciel"

On programme sous algobox un algorithme déjà écrit sous deux formes. Ce qui nous donne trois niveaux de langages pour un algorithme.

Après avoir écrit l'algorithme sous algobox, on leur demande d'en vérifier la cohérence avec un exemple.

Cette fiche n'a pas été testée en classe. Pour certains élèves, il faudrait sûrement prévoir d'autres exercices en plus.

Fiche 7 " L'instruction Si ... alors ..." Cette fiche n'a pas été testée.

Fiche 8 " Choisir les variables"

Certaines instructions sont réellement très difficiles à comprendre pour nos élèves : on les rencontre souvent à l'occasion des boucles mais on peut commencer à soulever le problème auparavant.

Il s'agit d'affectations du type : « N prend la valeur N + 1 » ou encore « S prend la valeur S + K »

Ici, il faut bien entendu comprendre que la *nouvelle valeur* de N s'obtient en ajoutant 1 à *l'ancienne valeur* de N.

Le but de la fiche 8 n'est pas d'apprendre à réduire systématiquement le nombre de variables d'un algorithme.

Par exemple, il ne nous viendrait pas à l'idée de désigner par une même lettre un nombre et son image par une fonction, cela risquerait d'être perturbant pour nos élèves.

Mais ces exercices sont là pour aider à se familiariser avec l'idée qu'une même variable peut prendre différentes valeurs selon l'état de ce qu'elle représente : valeur initiale/ valeur finale et parfois même il y aura des valeurs intermédiaires ...

L'idée est bien d'utiliser une même variable pour des paramètres de même nature (et derrière il y a la notion de suite de nombres).

Ainsi, utiliser une seule variable pour désigner un prix (initial) qui subit une variation et le prix final qui en résulte peut parfois être judicieux.

Fiche 9 " Boucle pour : initiation et applications"

Les situations 1 et 2 mettent en évidence le fait qu'il serait bien pratique de disposer d'outils évitant d'écrire plusieurs fois la même instruction (ou évitant d'écrire plusieurs fois des instructions, qui sans être identiques, se ressemblent beaucoup.)

L'instruction REPETER n FOIS n'existe que dans peu de langages informatiques mais elle a semblé intéressante ici : utiliser d'emblée POUR i ALLANT DE 1 à N semblait poser plus de difficultés dans la mesure où dans ce cas la variable i sert uniquement de compteur et n'est pas employée dans les instructions de la boucle (ce qui est troublant au départ).

Le fait que les deux situations peuvent se traiter avec la boucle pour est mis en évidence dans l'exercice 3 de la fiche applications.

Ces fiches ont été testées dans trois classes et dans deux d'entre elles le bilan est plutôt positif.

Fiche 10 " Tant que : initiation"

Cette fiche a été testée presque dans son intégralité, l'exercice 1 a été un peu modifié pour plus de clarté. Elle est longue : elle n'est pas conçue pour une séance d'une heure. Elle nécessite l'usage de l'ordinateur pour programmer sous algobox. Ce serait bien que les élèves aient le temps de programmer l'algorithme de l'exercice 3 **après l'avoir fait tourner à la main** : ils ont du mal à réaliser qu'ils doivent appliquer l'instruction plusieurs fois.

La difficulté du tant que est liée en grande partie à la condition d'arrêt pas toujours facile à interpréter et surtout plus tard à écrire, avec le risque d'un algorithme qui tourne indéfiniment.

Algorithmique : de quoi s'agit-il ?

Entrée	Traitement par une suite d'instructions	Sortie
	Pasteuriser le lait Maintenir à une température de 40° Incorporer les ferments lactiques Mettre en pot	
Ceci pourrait schématiser l'activité d'une entreprise de produits laitiers : chaque jour, entre dans l'entreprise une certaine quantité de lait, celui-ci subit un traitement et de cette entreprise sort une certaine quantité de yaourts.		

Dans notre vie de tous les jours, certaines de nos actions suivent le même modèle Entrée – Traitement – Sortie, ou tout au moins peuvent être « découpées » en un certain nombre d'instructions à appliquer dans un ordre précis.

En voici un exemple :

Préparation d'un flacon d'antibiotique à partir de « poudre pour suspension buvable » (lue sur la notice)

① Ouvrir le flacon en exerçant une forte pression sur le bouchon
② Retirer le film protecteur
③ Remplir le gobelet-doseur avec de l'eau jusqu'au trait
④ Verser la totalité du contenu du gobelet-doseur dans le flacon
⑤ Agiter vigoureusement

Exercice 1 Trouver deux « actions » de notre vie courante qui peuvent être découpées de la même façon en un certain nombre d'instructions

-
-

La démarche, qui consiste à appliquer **un certain nombre d'instructions dans un ordre précis**, est dite **algorithmique**. Elle est, depuis l'origine de l'histoire des sciences, une composante essentielle de l'activité mathématique.

Qu'est-ce qu'un algorithme ?

C'est une suite d'instructions à réaliser dans un ordre précis et dont le but est d'apporter une réponse à une question, de résoudre un problème (Quelle est la moyenne de ces trois nombres ? Quel est le plus grand diviseur commun de ces deux nombres ? ...)

En général, un <u>algorithme</u> est composé de trois étapes :	Exemple :
① **On entre des données** **C'est la phase d'ENTREE**	On entre 3 nombres
② **Ces données sont traitées par une suite d'instructions** **C'est la phase de TRAITEMENT**	① on calcule la somme des 3 nombres ② on divise cette somme par 3
③ **Un résultat est annoncé** **C'est la phase de SORTIE**	le résultat de la dernière opération est annoncé : c'est la moyenne des 3 nombres

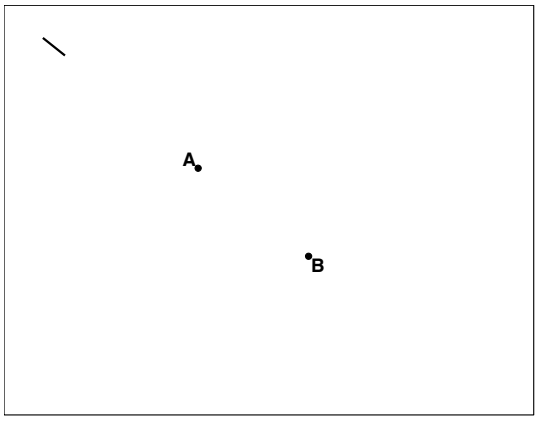
Lorsque la suite d'instructions est longue ou compliquée à appliquer et lorsque l'on doit l'utiliser souvent, il arrive fréquemment que l'on fasse appel à des machines (ordinateur, automates, ...) qui les font à notre place.

Il faut savoir aussi que les algorithmes ont beaucoup d'applications : dans le développement des logiciels informatiques, dans la gestion du trafic aérien, les téléphones portables ...

Exercice 2

a) Appliquer le traitement ci-dessous au segment dessiné.

Traitement
① Tracer un cercle dont le centre est une extrémité du segment et qui passe par l'autre extrémité du segment
② Même chose qu'au ① en prenant pour centre l'autre extrémité
③ Tracer la droite qui passe par les points d'intersection des deux cercles



b) Pour le segment [AB], que représente la droite qui a été tracée ? .....

Est-ce toujours le cas, quel que soit le segment donné au départ ? .....

c) Dans cet exemple, qu'a-t-on en entrée ? .....

Dans cet exemple, qu'a-t-on en sortie ? .....

ALGORITHMIQUE : Pour bien démarrer – Mémoires de la calculatrice

Mémoire courte de la calculatrice

On considère le nombre $A = 3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}$.

1) A l'aide de la calculatrice, donner une valeur approchée à 10^{-3} près du nombre A. .....

2) Toujours avec la calculatrice, donner une valeur approchée à 10^{-3} près de $3A$, ***sans retaper le nombre*** A. .....
Qu'a-t-il suffi de taper ? .....

3) ***Sans retaper le résultat précédent***, donner une valeur approchée à 10^{-3} près du nombre $\sqrt{3A}$. .....
Que suffit-il de taper ? .....

4) ***Sans tout retaper***, donner une valeur approchée à 10^{-3} près du nombre $\sqrt{3A} + \dfrac{1}{\sqrt{3A}}$. .....

Que suffit-il de taper ? .....

On vient donc d'obtenir une valeur approchée du nombre $\sqrt{3\times\left(3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}\right)} + \dfrac{1}{\sqrt{3\times\left(3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}\right)}}$

5) Justifier l'expression "mémoire courte". .....

ⓘ *La touche associée à la touche* [(–)] *permet de rappeler la réponse précédente*. [Ans] sur CASIO ou [rep] sur TI

Mémoires longues de la calculatrice

La calculatrice possède une mémoire composée de plusieurs "boîtes" qui portent chacune un nom et dans lesquelles on peut stocker des valeurs. Ces valeurs sont conservées même si la calculatrice est éteinte. Il suffit d'appeler la bonne boîte pour pouvoir utiliser le nombre qui s'y trouve.
Les noms des boîtes sont les lettres de l'alphabet. Le contenu d'une boîte peut changer ***mais pas son nom***.

Ainsi pour calculer $\sqrt{3A} + \dfrac{1}{\sqrt{3A}}$ avec $A = 3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}$, on peut commencer par mettre le nombre A dans une des boîtes de la calculatrice, par exemple la boîte A, puis faire le calcul comme indiqué ci-dessous.

① On tape $3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}$ sur la calculatrice

② On range ce nombre dans la boîte mémoire A [→] A pour CASIO ou [sto] A pour TI

③ On tape le calcul $\sqrt{3A} + \dfrac{1}{\sqrt{3A}}$ en utilisant la boite mémoire A

1) Vérifier qu'on obtient bien ainsi le même nombre qu'à la question **4)** du paragraphe précédent.

2) Stocker le résultat du calcul de $\sqrt{3A} + \dfrac{1}{\sqrt{3A}}$ dans une autre boîte mémoire, par exemple B.

3) En utilisant cette nouvelle boîte mémoire compléter (à 10^{-3} près). $\left(\sqrt{3A} + \dfrac{1}{\sqrt{3A}}\right)^2 + \dfrac{5}{\sqrt{3A} + \dfrac{1}{\sqrt{3A}}} + A \approx$ .....

Qu'a-t-il suffit de taper ? .....

Comme $A = 3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}$ ***on vient donc d'obtenir une valeur approchée du nombre***

$$\left(\sqrt{3\times\left(3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}\right)} + \dfrac{1}{\sqrt{3\times\left(3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}\right)}}\right)^2 + \dfrac{5}{\sqrt{3\times\left(3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}\right)} + \dfrac{1}{\sqrt{3\times\left(3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}\right)}}} + 3\times 5^4 + \dfrac{8\times 11^3}{2^3-1}$$

4) Quel est l'intérêt de mettre un nombre dans une boîte mémoire ? .....
.....

ALGORITHMIQUE : Mémoires de la calculatrice – Exercices

Exercice 1 :

En utilisant au mieux la calculatrice, donner la valeur exacte de $\dfrac{\frac{3}{5}-\frac{1}{7}\times 4^3+2}{\frac{3}{5}-\frac{1}{7}\times 4^3}+7\left(\frac{3}{5}-\frac{1}{7}\times 4^3\right)$. Expliquer votre méthode.

Exercice 2 :

Un élève vient de taper ces trois lignes sur sa calculatrice. ① $\frac{14}{3}\to A$ ② $3\times A+11\to B$ ③ $\sqrt{B}\to B$

1) ***Sans utiliser la calculatrice***, dire ce que contient la boîte mémoire B après avoir tapé les lignes ① et ②. ……

2) De même, dire ce que contient la boîte-mémoire B après avoir tapé les lignes ①, ② et ③. ……

Exercice 3 :

1) Taper la séquence ci-dessous sur la calculatrice, noter au fur et à mesure le résultat affiché par la calculatrice.

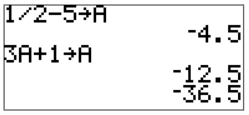
Séquence	$3\to A$ EXE entrer	$A^2\to A$ EXE entrer	EXE entrer	EXE entrer	EXE entrer
Affichage	3	……	……	……	……

2) De même, compléter le tableau ci-dessous. Comparer les résultats avec ceux du premier tableau.

Séquence	$3\to A$ EXE entrer	A^2 EXE entrer	EXE entrer	EXE entrer	EXE entrer
Affichage	3	……	……	……	……

Exercice 4 :

Voici un écran de calculatrice. Expliquer le dernier résultat affiché.



Exercice 5 :

La première ligne du tableau ci-dessous donne une séquence de calcul tapée sur une calculatrice. La ligne suivante présente les réponses affichées par la calculatrice. Compléter les pointillés ***sans utiliser la calculatrice***. Vérifier ensuite avec la calculatrice.

Séquence	$\sqrt{2}+3\to A$ EXE entrer	$A^2-6A\to B$ EXE entrer	$2B\to B$ EXE entrer	EXE entrer	EXE entrer
Affichage	4.414213562	……	……	……	……

Synthèse

- L'écriture $4\to A$ signifie que l'on …… ***la valeur 4 dans la mémoire*** **A**.
- On peut dire aussi **A *prend la valeur 4*** ou encore ***Mettre 4 dans*** **A**

ALGORITHMIQUE : Pour bien démarrer – Mémoires de la calculatrice

Comprendre la mise en mémoire pour mieux comprendre plus tard la notion de variable
Pour valider une action (exécution d'un calcul par exemple) utiliser la touche [EXE] pour CASIO ou la touche [entrer] pour TI
A chaque touche correspond une ou deux autres actions repérées par des couleurs. Pour y accéder, taper d'abord sur la touche de même couleur

Mémoire courte de la calculatrice

On considère le nombre $A = 3\times5^4 + \dfrac{8\times11^3}{2^3-1}$.

Touche [∧] pour taper une puissance

1) A l'aide de la calculatrice, donner une valeur approchée à 10^{-3} près du nombre A. ≈ 3 396,143

2) Toujours avec la calculatrice, donner une valeur approchée à 10^{-3} près de 3A, ***sans retaper le nombre*** A. ≈ 10 188,429
Qu'a-t-il suffi de taper ? La valeur de A est restée affichée ⇒ Il suffit de taper ×3

3) ***Sans retaper le résultat précédent***, donner une valeur approchée à 10^{-3} près du nombre $\sqrt{3A}$. ≈ 100,938
Que suffit-il de taper ? Le résultat obtenu au **2)** (valeur de 3A) est resté affiché ⇒ Il suffit de taper [√] [Ans] ou [√] [rep]

4) ***Sans tout retaper***, donner une valeur approchée à 10^{-3} près du nombre $\sqrt{3A} + \dfrac{1}{\sqrt{3A}}$. ≈ 100,948

Que suffit-il de taper ? Le résultat obtenu au **3)** (valeur de $\sqrt{3A}$) est resté affiché ⇒ [Ans] + 1 ÷ [Ans] ou [rep] + 1 ÷ [rep]

On vient donc d'obtenir une valeur approchée du nombre $\sqrt{3\times\left(3\times5^4+\dfrac{8\times11^3}{2^3-1}\right)}+\dfrac{1}{\sqrt{3\times\left(3\times5^4+\dfrac{8\times11^3}{2^3-1}\right)}}$

5) Justifier l'expression "mémoire courte". Ne conserve que le dernier résultat
De plus, résultat perdu quand on efface l'écran (selon calculatrices)

ⓘ *La touche associée à la touche* [(–)] *permet de rappeler la réponse précédente.* [Ans] sur CASIO ou [rep] sur TI

Mémoires longues de la calculatrice

La calculatrice possède une mémoire composée de plusieurs "boîtes" qui portent chacune un nom et dans lesquelles on peut stocker des valeurs. Ces valeurs sont conservées même si la calculatrice est éteinte. Il suffit d'appeler la bonne boîte pour pouvoir utiliser le nombre qui s'y trouve.
*Les noms des boîtes sont les lettres de l'alphabet. Le contenu d'une boîte peut changer **mais pas son nom**.*

Ainsi pour calculer $\sqrt{3A} + \dfrac{1}{\sqrt{3A}}$ avec $A = 3\times5^4 + \dfrac{8\times11^3}{2^3-1}$, on peut commencer par mettre le nombre A dans une des boîtes de la calculatrice, par exemple la boîte A, puis faire le calcul comme indiqué ci-dessous.

① On tape $3\times5^4 + \dfrac{8\times11^3}{2^3-1}$ sur la calculatrice

② On range ce nombre dans la boîte mémoire A en tapant [→] A pour CASIO ou [sto] A pour TI

③ On tape le calcul $\sqrt{3A} + \dfrac{1}{\sqrt{3A}}$ en utilisant la boite mémoire A

1) Vérifier qu'on obtient bien ainsi le même nombre qu'à la question **4)** du paragraphe précédent. Réponse positive obligatoire

2) Stocker le résultat du calcul de $\sqrt{3A} + \dfrac{1}{\sqrt{3A}}$ dans une autre boîte mémoire, par exemple B. [→] B ou [sto] B

3) En utilisant cette nouvelle boîte mémoire compléter (à 10^{-3} près). $\left(\sqrt{3A} + \dfrac{1}{\sqrt{3A}}\right)^2 + \dfrac{5}{\sqrt{3A} + \dfrac{1}{\sqrt{3A}}} + A \approx$ 13 586,621

Qu'a-t-il suffit de taper ? Avec la lettre B l'expression s'écrit $B^2 + \dfrac{5}{B} + A$ ⇒ On tape B [x^2] + 5 ÷ B + A

Comme $A = 3\times5^4 + \dfrac{8\times11^3}{2^3-1}$ ***on vient donc d'obtenir une valeur approchée du nombre***

$$\left(\sqrt{3\times\left(3\times5^4+\frac{8\times11^3}{2^3-1}\right)}+\frac{1}{\sqrt{3\times\left(3\times5^4+\frac{8\times11^3}{2^3-1}\right)}}\right)^2+\frac{5}{\sqrt{3\times\left(3\times5^4+\frac{8\times11^3}{2^3-1}\right)}+\frac{1}{\sqrt{3\times\left(3\times5^4+\frac{8\times11^3}{2^3-1}\right)}}}+3\times5^4+\frac{8\times11^3}{2^3-1}$$

4) Quel est l'intérêt de mettre un nombre dans une boîte mémoire ? Nombre conservé, même si on éteint la calculatrice ou si on efface l'écran
Evite de taper plusieurs fois le même nombre (fastidieux, surtout quand le nombre est compliqué, avec en prime, plus de risques d'erreurs de frappe)

ALGORITHMIQUE : Mémoires de la calculatrice – Exercices

Pour avoir cette fiche les élèves doivent d'abord valider leur travail sur la ficheprécédente en m'appelant

Exercice 1 : Résultat sous forme de fraction : Touche [F↔D] une fois le calcul affiché pour CASIO, choisir ▶Frac dans le menu [math] pour TI

En utilisant au mieux la calculatrice, donner la valeur exacte de $\dfrac{\frac{3}{5}-\frac{1}{7}\times 4^3+2}{\frac{3}{5}-\frac{1}{7}\times 4^3}+7\left(\frac{3}{5}-\frac{1}{7}\times 4^3\right)$. Expliquer votre méthode.

"Utiliser au mieux" signifie qu'il faut utiliser sa calculatrice le plus efficacement possible, donc éviter de répéter plusieurs fois la même chose

Le nombre $C=\frac{3}{5}-\frac{1}{7}\times 4^3$ apparaît 3 fois, avec cette lettre le calcul devient $\frac{C+2}{C}+7C$ ⇒ On tape $\frac{3}{5}-\frac{1}{7}\times 4^3 \to C$ puis $(C+2)\div C+7C$ $-\frac{88\,256}{1\,495}$

ⓘ Nouvelle CASIO : La touche [□/□] peut remplacer la touche de division, les parenthèses sont alors inutiles et le résultat est directement sous forme de fraction

Exercice 2 :

Un élève vient de taper ces trois lignes sur sa calculatrice. ① $\frac{14}{3}\to A$ ② $3\times A+11\to B$ ③ $\sqrt{B}\to B$

$3\times\frac{14}{3}+11\to B$ $\sqrt{25}\to B$

1) ***Sans utiliser la calculatrice***, dire ce que contient la boîte mémoire B après avoir tapé les lignes ① et ②. $14+11=25$

2) De même, dire ce que contient la boîte-mémoire B après avoir tapé les lignes ①, ② et ③. $\sqrt{25}=5$

Exercice 3 :

1) Taper la séquence ci-dessous sur la calculatrice, noter au fur et à mesure le résultat affiché par la calculatrice.

Séquence	$3\to A$ EXE entrer	$A^2\to A$ EXE entrer	EXE entrer	EXE entrer	EXE entrer
Affichage	3	9	81	6561	43 046 721

A chaque frappe de [EXE] ou [entrer] on calcule le carré du nombre contenu dans la boîte A, puis on range le résultat à nouveau dans A. Le contenu de la boîte A change donc à chaque fois.

A chaque frappe, le contenu de la boîte A est remplacépar le carré du nombre précédent

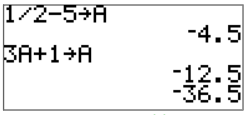
2) De même, compléter le tableau ci-dessous. Comparer les résultats avec ceux du premier tableau.

Séquence	$3\to A$ EXE entrer	A^2 EXE entrer	EXE entrer	EXE entrer	EXE entrer
Affichage	3	9	9	9	9

Le résultat du calcul du carré du nombre contenu dans la boîte A n'a pas été stocké, la calculatrice affiche donc toujours le même résultat.

Exercice 4 :

Voici un écran de calculatrice. Expliquer le dernier résultat affiché.



La calculatrice a calculé $3\times(-4,5)+1$
La calculatrice a calculé $3\times(-12,5)+1$

ⓘ Comme au **1)** de l'exercice précédent, taper [EXE] ou [entrer] après l'affichage de -12,5 a provoqué à nouveau le calcul de 3 A + 1

Exercice 5 :

La première ligne du tableau ci-dessous donne une séquence de calcul tapée sur une calculatrice. La ligne suivante présente les réponses affichées par la calculatrice. Compléter les pointillés ***sans utiliser la calculatrice***. Vérifier ensuite avec la calculatrice.

Séquence	$\sqrt{2}+3\to A$ EXE entrer	$A^2-6A\to B$ EXE entrer	$2B\to B$ EXE entrer	EXE entrer	EXE entrer
Affichage	4.414213562	-7	$-7\times 2=-14$	$-14\times 2=-28$	$-28\times 2=-56$

Identité $(a+b)^2=a^2+2ab+b^2$

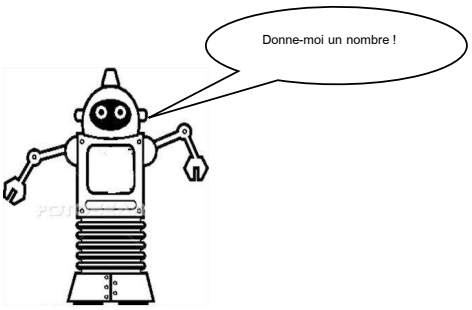
$(\sqrt{2}+3)^2-6(\sqrt{2}+3)=11-18$
$=2+6\sqrt{2}+9-6\sqrt{2}-18$
$=11-18$

Synthèse

- L'écriture [$4\to A$] signifie que l'on ***stocke la valeur 4 dans la mémoire A***. Stocker/Ranger
- On peut dire aussi [**A *prend la valeur* 4**] ou encore [***Mettre* 4 *dans* A**] Affecter 4 à A

Algorithmique (2) : Des suites d'instructions et des robots

Exercice 1



On imagine qu'on dispose de deux robots.

Chacun d'entre eux est conçu pour pouvoir appliquer des instructions du type :

- demander à l'utilisateur du robot de donner un nombre
- mémoriser un nombre (mémoire longue)
- faire des calculs (ajouter, multiplier, calculer le carré d'un nombre)
- retenir le résultat du dernier calcul (mémoire courte)
- annoncer un résultat

On donne au robot 1 la suite d'instructions suivante	On donne au robot 2 la suite d'instructions suivante
① **Demander à l'utilisateur de donner un nombre** ② **Mémoriser le nombre donné** ③ **Ajouter - 2 au nombre mémorisé** ④ **Calculer le carré du résultat précédent** ⑤ **Ajouter 1 au résultat précédent** ⑥ **Annoncer le dernier résultat à l'utilisateur**	① **Demander à l'utilisateur de donner un nombre** ② **Mémoriser le nombre donné** ③ **Ajouter - 4 au nombre mémorisé** ④ **Multiplier le résultat précédent par le nombre mémorisé** ⑤ **Ajouter 5 au résultat précédent** ⑥ **Annoncer le dernier résultat à l'utilisateur**

a) On met le robot 1 en marche. On lui donne le nombre 5. Quel résultat le robot 1 va-t-il annoncer ? ...

b) On met le robot 2 en marche. On lui donne aussi le nombre 5. Quel résultat le robot 2 va-t-il annoncer ? ...

c) Donne aux deux robots le nombre de ton choix :

Nombre entré : ... Annonce du robot 1 : ... Annonce du robot 2 : ...

d) Prouve que les deux robots annoncent toujours le même résultat quand on leur donne le même nombre.

.....

.....

.....

Exercice 2 On considère la fonction définie par : $f(x) = 5(x^2 + x)$

a) Calculer l'image de -1 et celle de 4 par f : .....

b) On dispose d'un robot, identique à ceux de l'exercice 1.

Complète la suite d'instructions ci-dessous pour que le robot calcule l'image de ce nombre par la fonction f et annonce le résultat à l'utilisateur

① **Demander à l'utilisateur de donner un nombre**

② **Mémoriser le nombre donné**

③ .....

...

Exercice 3 Même exercice avec $f(x) = \frac{3}{x-2}$ sachant que le robot peut aussi calculer l'inverse d'un nombre.

① **Demander à l'utilisateur de donner un nombre** ④.....

② **Mémoriser le nombre donné** ⑤.....

③ ..... ⑥.....

Algorithmique (3) : Des suites d'instructions et des boîtes-mémoires

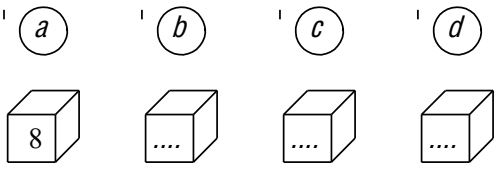
Evidemment nous n'emploierons pas de robot comme dans la fiche précédente mais plus tard, nous serons amenés à donner des suites d'instructions à un ordinateur ou à la calculatrice ...
On redonne ci-dessous la suite d'instructions du robot 2 :

① **Demander à l'utilisateur de donner un nombre**
② **Mémoriser le nombre donné**
③ **Ajouter - 4 au nombre mémorisé**
④ **Multiplier le résultat précédent par le nombre mémorisé**
⑤ **Ajouter 5 au résultat précédent**
⑥ **Annoncer le dernier résultat à l'utilisateur**

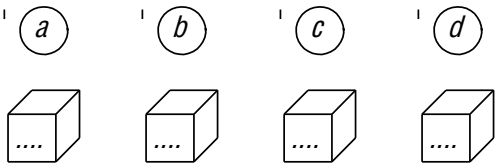
Pour « mémoriser » le nombre entré par l'utilisateur, on peut imaginer qu'il est stocké dans une «boîte-mémoire».
De même pour les résultats des calculs des instructions ③ , ④ et ⑤ .
Ainsi, pour cet algorithme, on peut considérer que quatre boîtes vont être utilisées :

- la boîte *a* : pour le nombre entré par l'utilisateur
- les boîtes *b*, *c* et *d* : pour chacun des résultats des calculs des instructions ③ , ④ et ⑤ .

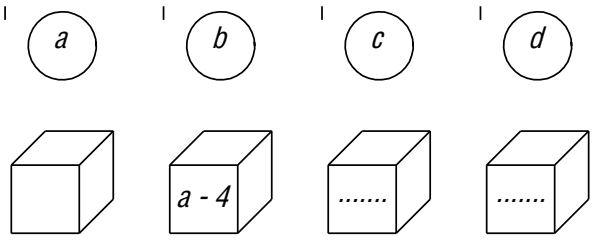
a) Dans cette question, on suppose que l'utilisateur entre la valeur 8.
Ainsi la boîte-mémoire *a* contient maintenant la valeur 8. Compléter les autres boîtes-mémoires par leur contenu:



b) Recommencer la question précédente en supposant que le nombre entré est 2 :



c) Compléter les pointillés **<u>en utilisant à chaque fois le nom de la boîte précédente</u>** (et une autre si nécessaire)



d) L'utilisation de ces boîtes va permettre de réécrire les instructions sous une autre forme, plus proche des instructions spécifiques aux outils (ordinateur ou calculatrice) que nous utiliserons.
Ainsi l'instruction ③ : « **Ajouter - 4 au nombre mémorisé** » devient : « **mettre *a* – 4 dans *b*** ».
Ce qu'il faut comprendre de la façon suivante : on soustrait 4 à la valeur qui est dans *a* et on met le résultat de cette soustraction dans *b*.
On peut formuler autrement et dire par exemple : « ***b* prend la valeur *a* – 4** » ou « **remplacer *b* par *a* – 4** ».
Compléter les .... ci-dessous pour dire ce que deviennent les instructions ④ et ⑤ avec cette façon d'écrire.
④ **Multiplier le résultat précédent par le nombre mémorisé** devient : « **mettre** ..... **dans** .... »
⑤ **Ajouter 5 au résultat précédent** devient : « .....»

e) Les instructions ① et ② peuvent être résumées en une seule comme le montre le tableau ci-dessous.

Compléter les ... des instructions ❸ , ❹ et ❺ (pour ❺, on attend un verbe en rapport avec celui de l'instruction ❶.)

Instructions écrites en langage courant	Ecriture proche de celles des ordinateurs et calculatrices	**Commentaires**
		Cette écriture n'est pas «figée», elle peut varier selon les manuels utilisés. C'est une écriture intermédiaire entre le langage courant et les différents langages de programmation sur ordinateur ou calculatrice.
① **Demander à l'utilisateur de donner un nombre**	❶ Entrer *a*	L'instruction **Entrer *a*** signifie trois choses : 1) elle permet de demander à l'utilisateur d'entrer une valeur 2) elle définit le nom de la boîte dans laquelle va être rangée la réponse de l'utilisateur 3) elle stocke la valeur entrée par l'utilisateur dans la boîte *a.* *Dans certains manuels, on utilise plutôt l'instruction « Saisir a »* *On rencontrera aussi l'expression «Lire a», qui aura le même sens.*
② **Mémoriser le nombre donné**		
③ **Ajouter - 4 au nombre mémorisé**	❷ Mettre *a* – 4 dans *b*	Chacun de ces calculs s'effectue en utilisant un nombre déjà rangé dans une boîte. Chaque résultat est rangé dans une autre boîte. « Mettre *a* – 4 dans *b* » signifie deux choses : • on soustrait 4 à la valeur qui est dans *a* • on place le résultat dans *b*
④ **Multiplier le résultat précédent par le nombre mémorisé**	❸ Mettre ..... dans *c*	
⑤ **Ajouter 5 au résultat précédent**	❹ Mettre ..... dans *d*	
⑥ **Annoncer le dernier résultat A l'utilisateur**	❺ ..... *d*	La valeur contenue dans la boîte *d* est annoncée. (attention, ce n'est pas *la lettre d* qui est annoncée) *On rencontrera aussi l'expression « Afficher d » qui aura le même sens.*

Application On redonne ci-dessous la liste d'instructions du robot 1 de la fiche précédente.

① Demander à l'utilisateur de donner un nombre
② Mémoriser le nombre donné
③ Ajouter - 2 au nombre mémorisé
④ Calculer le carré du résultat précédent
⑤ Ajouter 1 au résultat précédent
⑥ Annoncer le dernier résultat à l'utilisateur

Ecrire les instructions comme on vient de le faire dans le tableau de la question e).

❶
❷

En fait on aurait pu n'utiliser que deux boîtes-mémoires : une pour la valeur entrée par l'utilisateur et l'autre pour la réponse à annoncer. Ecrire ci-dessous l'instruction 2 pour que l'algorithme sorte la réponse que donne le robot 1.

❶ Entrer *a*
❷
❸ Sortir *b*

Algorithmique (4) : Des suites d'instructions et des variables

Dorénavant, nous ne dirons plus « **boîte-mémoire *a*** » mais nous parlerons de la « **variable *a*** ».

Exercice 1

a) On propose ci-dessous un algorithme. Compléter les … .

Algorithme
entrer x
mettre $x - 4$ dans a
mettre $2 \times a$ dans b
mettre $b + 3$ dans c
sortir c

Contenu des **variables** ligne par ligne			
$x = 3$			
$x =$ …	$a =$ …		
$x =$ …	$a =$ …	$b =$ …	
$x =$ …	$a =$ …	$b =$ …	$c =$ ...

Quelle valeur est annoncée ? …

b) Que sort cet algorithme si on entre - 8 ? **……**

c) Compléter cette autre écriture du même algorithme :

⇨ choisir un nombre
⇨ soustraire 4 au nombre choisi
⇨ **……**
⇨ **……**
⇨ **……**

Exercice 2

On considère l'algorithme ci-contre :

entrer a
mettre $3 \times a - 1$ dans a
Sortir a

Que sort cet algorithme si on entre 5 ? …… Et si on entre - 3 ? ……

Exercice 3

On considère les quatre algorithmes ci-dessous.

Algorithme 1	Algorithme 2	Algorithme 3	Algorithme 4
Entrer n Mettre $2n$ dans a Mettre $a + 3$ dans b Sortir b	Entrer n Mettre $2n$ dans a Mettre $a + 3$ dans a Sortir a	Entrer n Mettre $2n + 3$ dans a Sortir a	Entrer n Mettre $2n + 3$ dans n Sortir n

a) Ecrire sous chaque algorithme la liste de toutes les variables utilisées.

b) Pour chaque algorithme, donner ci-dessous la valeur sortie si on entre 5 .

Algorithme 1	Algorithme 2	Algorithme 3	Algorithme 4

Que remarque-t-on ? ……

La remarque précédente reste-t-elle vraie quelle que soit la valeur entrée ? ……

Justifier. ……

Algorithmique (5) :
Trouver le but d'un algorithme, écrire ou modifier un algorithme

Exercice 1

On considère l'algorithme ci-dessous

Dans la première ligne, on a écrit la liste de toutes les variables utilisées par la suite.

Variables a , b , c , x
Début
entrer a
entrer b
entrer c
x prend la valeur $\frac{a+b+c}{3}$
sortir x
Fin

a) Si on entre les valeurs 5 , 8 et 11 , quelle valeur sort cet algorithme ? .....

b) Que représente la valeur sortie par rapport aux trois valeurs entrées ?

.....

Le but de cet algorithme est donc .....

Exercice 2

Un professeur a donné trois devoirs à ses élèves au 1er trimestre. Comme le dernier devoir a duré 2 heures, il lui affecte le coefficient 2 , tandis que les deux autres ont pour coefficient 1.

a) Etienne a eu 12 et 10 aux deux premiers devoirs et 15 au dernier. Quelle est sa moyenne ? .....

b) Proposer ci-dessous un algorithme qui :

- demande à l'utilisateur d'entrer successivement 3 nombres, correspondant aux trois notes d'un élève (on considérera que la dernière entrée est celle du dernier devoir).
- affiche la moyenne trimestrielle de cet élève.

Variables
Début
Fin

Exercice 3

1° Lire et comprendre l'algorithme ci-contre :

2° La phrase suivante est-elle vraie ?

« Cet algorithme permet d'échanger les valeurs de A et de B »

3° Compléter le tableau ci-dessous par les valeurs des variables A et B.

	A	B
Valeurs entrées par l'utilisateur	7	3
Valeurs après l'instruction ③		
Valeurs après l'instruction ④		

Variables
A , B
Début
① Entrer A
② Entrer B
③ Mettre B dans A
④ Mettre A dans B
⑤ Afficher " A vaut : "
⑥ Sortir A
⑦ Afficher " B vaut : "
⑧ Sortir B
Fin

4° Compléter l'algorithme ci-dessous de sorte qu'il échange les valeurs de A et de B en utilisant uniquement le vocabulaire utilisé dans l'algorithme précédent.

Variables

Début
Entrer A
Entrer B

Afficher " A vaut : "
Sortir A
Afficher " B vaut : "
Sortir B
Fin

Exercice 4

Quelques amis projettent d'organiser un séjour d'une semaine à la montagne. La location de l'appartement coûte 600 euros et le forfait hebdomadaire pour les remontées mécaniques est de 200 euros par skieur. L'appartement permet d'héberger jusqu'à 10 personnes. Ils ne savent pas encore combien de personnes participeront à ce séjour.

a) Quel sera le coût total (location + forfait) pour chacun des participants s'ils partent à 5 ? ……

S'ils partent à 8 ? …….

b) Proposer un algorithme qui :

- demande d'entrer le nombre de participants
- sort le coût par personne.

Algorithmique (6) : Utiliser un logiciel de programmation

Exercice 1

On a déjà rencontré l'algorithme ci-dessous écrit sous deux formes.

On introduit une nouvelle écriture qui correspond à celle du logiciel que l'on va utiliser :

Instructions écrites en langage courant	Ecriture proche de celles des ordinateurs et calculatrices	Ecriture sous le logiciel *algobox*
① Demander à l'utilisateur de donner un nombre.	❶ Entrer a	VARIABLES a EST_DU_TYPE NOMBRE b EST_DU_TYPE NOMBRE c EST_DU_TYPE NOMBRE d EST_DU_TYPE NOMBRE DEBUT_ALGORITHME LIRE a b PREND_LA_VALEUR a-4 c PREND_LA_VALEUR b*a d PREND_LA_VALEUR c+5 AFFICHER d FIN_ALGORITHME
② Ajouter - 4 au nombre donné	❷ mettre $a - 4$ dans b	
③ Multiplier le résultat précédent par le nombre donné	❸ mettre $b \times a$ dans c	
④ Ajouter 5 au résultat précédent	❹ mettre $c + 5$ dans d	
⑤ Annoncer le dernier résultat à l'utilisateur	❺ Sortir d	

a) Comparer les écritures des 2ème et 3ème colonnes : .....

.....

b) Ouvrir le logiciel *algobox* et écrire cet algorithme.

c) Lancer l'algorithme et entrer une valeur.

Vérifier la cohérence du résultat : .....

Exercice 2

On a écrit ci-dessous un algorithme sous *algobox*. Le traduire en langage courant.

```
1     VARIABLES
2       x EST_DU_TYPE NOMBRE
3       y EST_DU_TYPE NOMBRE
4       z EST_DU_TYPE NOMBRE
5     DEBUT_ALGORITHME
6       LIRE x
7       y PREND_LA_VALEUR x+2
8       z PREND_LA_VALEUR 5*y
9       AFFICHER z
10    FIN_ALGORITHME
```

Exercice 3

Reprendre l'exercice 4 de la fiche 5 .

Ecrire l'algorithme sous *algobox* puis lancer le programme pour compléter le tableau ci-dessous :

Nombre de participants	1	2	3	4	5	6	7	8	9	10
Coût par participant										

Algorithmique (7) :
L'instruction « Si … alors … »

Exercice 1 On considère l'algorithme ci-dessous :

Variables
X , Y
Début
Entrer X
Si $X > 5$ alors mettre $2X + 5$ dans Y
Si $X = 5$ alors mettre 12 dans Y
Si $X < 5$ alors mettre $8 - 3X$ dans Y
Sortir Y
Fin

Compléter :

Valeur entrée	3,2	10	5	7	2
Valeur annoncée					

Exercice 2

Les amis qui souhaitent organiser un séjour aux sports d'hiver ont appris que pour des groupes de plus de 6 personnes le forfait hebdomadaire pour les remontées mécaniques s'élève à 180 € au lieu de 200 €.

On rappelle que le coût de location de l'appartement se monte à 600 €.

Proposer un algorithme qui demande d'entrer le nombre de participants au séjour et qui annonce le prix total par participant.

Exercice 3 On considère l'algorithme ci-dessous.

Variables
p , X
Début
entrer X
si X est inférieur ou égal à 250 alors mettre 98 dans p
sinon mettre $98 + (X - 250) \times 0{,}34$ dans p
sortir p
Fin

1° Quelle valeur sort cet algorithme si on entre 123 ? ……

2° Quelle valeur sort cet algorithme si on entre 280 ? …… Si on entre 250 ? ……. Si on entre 330 ? ……

3° L'agence de location de voitures *Autoloc* possède un logiciel permettant de calculer le coût p (en €) de location d'un véhicule pour X kilomètres parcourus. Le logiciel utilise pour cela l'algorithme décrit ci-dessus.

En déduire les informations manquantes dans l'encadré ci-dessous :

Autoloc
Tarif de location : … €
Ce tarif permet de parcourir … kilomètres.
Il faut compter … € par kilomètre supplémentaire.

Exercice 4 Une entreprise de transport possède 4 cars de 50 places chacun et se propose d'assurer le transport des supporters d'une équipe de rugby. Chaque car se loue 800 € tout compris.

1° Quel est le nombre maximal de supporters que cette entreprise peut transporter ? ……

2° On suppose que 50 supporters se rendent au stade. Quel est le coût pour chacun d'entre eux ? ……

3° Même question avec 160 supporters : …… . Puis avec 120 supporters : ……

4° Expliquer comment déterminer le coût par personne quand on connaît le nombre N de supporters.

……

……

……

……

……

5° Proposer ci-dessous un algorithme qui affiche le coût par personne une fois qu'on a entré le nombre de supporters à transporter.

Algorithmes (8) : Choisir les variables

Exercice 1

Un professeur demande à ses élèves d'écrire un algorithme qui :

- demande d'entrer un nombre, qui, pour l'utilisateur, représente un prix en euros.
- calcule et affiche le nouveau prix après avoir appliqué une réduction de 30 %

a) Etienne, un élève, a écrit l'algorithme ci-contre :

Dire ce que représentent p_1, p_2 et r dans le contexte de l'exercice :

……

……

……

Si on entre la valeur 120, quel nombre affiche l'algorithme d'Etienne ? .....

Son algorithme semble-t-il correct ? .....

Variables
p_1, p_2 et r
Début
① Afficher « Donnez le prix initial »
② Entrer p_1
③ r prend la valeur $p_1 \times 0{,}3$
④ p_2 prend la valeur $p_1 - r$
⑤ Afficher « Le prix après réduction est : »
⑥ Afficher p_2
Fin

b) La voisine d'Etienne, Marie, fait la remarque suivante : « Pourquoi prendre trois variables pour écrire l'algorithme ? Je n'en ai utilisé que deux. »

Compléter l'algorithme de Marie :

Variables
p_1, p_2
Début
① Afficher « Donnez le prix initial »
②
③
④ Afficher « Le prix après réduction est : »
⑤
Fin

c) Quant à Tom, un autre élève, il a réussi à écrire un algorithme qui fonctionne en utilisant une seule variable. Compléter l'algorithme de Tom :

Variables
p
Début
① Afficher « Donnez le prix initial »
②
③
④ Afficher « Le prix après réduction est : »
⑤
Fin

Exercice 2 Lors d'un concours, 40 candidats ont passé une épreuve de mathématiques.

Les copies ont été réparties dans deux paquets : le paquet A a été corrigé par M. X et le paquet B par M. Y.

La moyenne des notes attribuées par M. X est 8 et celle des notes attribuées par M. Y est 11.

Par souci d'équité, le jury a décidé de modifier les notes du paquet A de la façon suivante :

on augmente toutes les notes de 25 % et on ajoute 1 point au résultat obtenu.

Proposer un algorithme qui :

- utilise le moins de variables possible
- demande d'entrer une note du paquet A
- affiche la nouvelle note obtenue après modification

Algorithmique (9) : Boucle Pour (initiation)

Situation 1

Une observation faite sur la fréquentation d'un stade de football a permis de constater que:

- chaque année, trois quarts des abonnés de l'année précédente se réabonnent.
- chaque année, il y a 2 500 nouveaux abonnés.

On suppose que la situation décrite par l'observation reste la même au fil des ans et permet d'obtenir une estimation satisfaisante du nombre d'abonnés dans les années suivantes.

1° On suppose qu'en 2010, il y avait 8000 abonnés. Combien d'abonnés devrait compter ce stade en 2011 ?

.....

Combien d'abonnés devrait compter ce stade en 2012 ?.....

2° On considère l'algorithme ci-contre.

a) Qu'affiche l'algorithme si l'utilisateur entre la valeur 8000 ?

.....

b) Si on arrondit à l'unité la valeur affichée, que représente-t-elle dans le contexte de l'exercice ?

.....

.....

Variables
p est un nombre
Début
Afficher « Quel est le nombre d'abonnés en 2010 ? »
Lire *p*
p prend la valeur *p* × 0,75 + 2500
p prend la valeur *p* × 0,75 + 2500
p prend la valeur *p* × 0,75 + 2500
p prend la valeur *p* × 0,75 + 2500
Afficher *p*
Fin

3° Que faut-il ajouter à cet algorithme pour qu'il affiche une estimation du nombre d'abonnés en 2020 ?

.....

Ecrire 10 fois la même instruction revient à dire de la répéter 10 fois (voir algorithme ci-contre)

Variables
p est un nombre
Début
Afficher « Quel est le nombre d'abonnés en 2010 ? »
Lire *p*
Répéter 10 fois :
- *p* prend la valeur *p* × 0,75 + 2500

Afficher *p*
Fin

4° Compléter l'algorithme ci-contre de sorte qu'il :

- demande à l'utilisateur d'entrer un nombre entier *n*
- affiche le nombre d'abonnés que compte ce stade au bout de *n* années (après 2010).

Variables

Début
Afficher « Quel est le nombre d'abonnés en 2010 ? »
Lire *p*

Situation 2

Le service « abonnement » d'une revue locale mensuelle, dont le premier numéro a paru en 2000, a fait les observations suivantes :

- d'une année à l'autre, seuls 7 abonnés sur 10 se réabonnent.
- quant aux nouveaux abonnements, on note que : en 2001, ils étaient au nombre de 100 ; en 2002, ils s'élevaient à 200, pour atteindre 300 en 2003. Et ainsi de suite les années suivantes : le nombre de nouveaux abonnés augmente de 100 par an.

1° On considère l'algorithme ci-contre.
Le compléter pour qu'il affiche le nombre d'abonnés en 2005.

2° Expliquer pourquoi on ne peut pas utiliser ici l'instruction « Répéter » .
.....
.....

Variables
p est un nombre
Début
Afficher « Quel est le nombre d'abonnés en 2000 ? »
Lire *p*
p prend la valeur ....
p prend la valeur ....

Afficher *p*
Fin

3° Quel est le nombre de nouveaux abonnés *n* années après 2000 (c'est-à-dire en l'an 2000 + *n*) ? .....
Si on note *p* le nombre total d'abonnés de l'année précédente, exprimer en fonction de *n* et de *p* le nombre total d'abonnés en l'an 2000 + *n* : .....

Ainsi, il suffit de dire d'appliquer l'instruction «p prend la valeur p× 0,7+100 ×n » en changeant les valeurs de n .
Voici comment cela se traduit :

Variables
p et *n* sont des nombres
Début
Afficher « Quel est le nombre d'abonnés en 2000 ? »
Lire *p*
Pour n allant de 1 à 5 *:*
▪ *p prend la valeur p* × 0,7 + 100 × *n*
Afficher *p*

Il faut comprendre que la variable n change de valeur : n prend successivement toutes les valeurs entières de 1 à 5 .
Et à chaque fois que n change de valeur, on recommence à appliquer l'instruction « p prend la valeur p × 0,7 + 100 × *n ».*

4° Compléter l'algorithme ci-contre de sorte qu'il :

- demande à l'utilisateur d'entrer un nombre entier *M*
- affiche le nombre d'abonnés au bout de *M* années (après 2000).

Variables

Début
Afficher « Quel est le nombre d'abonnés en 2000 ? »
Lire *p*

Algorithmique (9) : Boucle pour (applications)

Exercice 1

On considère l'algorithme 1 ci-contre.

a) Quel est le but de cet algorithme ?

.....

b) Proposer ci-dessous un algorithme qui :

- a le même but que l'algorithme 1
- est beaucoup plus court que l'algorithme 1.

Algorithme 1

Variables

p est un nombre

Début

p prend la valeur 10 × 10
afficher *p*
p prend la valeur 11 × 11
afficher *p*
p prend la valeur 12 × 12
afficher *p*
p prend la valeur 13 × 13
afficher *p*
p prend la valeur 14 × 14
afficher *p*
p prend la valeur 15 × 15
afficher *p*
p prend la valeur 16 × 16
afficher *p*
p prend la valeur 17 × 17
Afficher *p*
p prend la valeur 18 × 18
afficher *p*
p prend la valeur 19 × 19
Afficher *p*
p prend la valeur 20 × 20
Afficher *p*

Fin

Algorithme 2

Variables

Début

Fin

Exercice 2

On considère l'algorithme ci-contre.

On suppose que l'utilisateur entre la valeur 5

a) Combien de fois l'instruction « *S* prend la valeur *S + i* » va-t-elle être appliquée ? ...

b) Compléter le tableau ci-dessous.

Variables

S, i, n sont des nombres

Début

① S prend la valeur 0
② Lire *n*
③ Pour *i* allant de 1 à *n*, *S* prend la valeur *S + i*
④ Afficher S

Fin

Valeurs des variables	*n*	*i*	*S*
Après l'instruction ②			
Puis après avoir appliqué une fois la ligne ③			
...			
…			

c) Qu'affiche cet algorithme si la valeur entrée est 5? .....

d) Sans recommencer dire ce qu'afficherait cet algorithme si la valeur entrée était 7.

e) Quel est le but de cet algorithme ? .....

Exercice 3

On considère l'algorithme ci-contre**.**

Quelle valeur affiche-t-il ? .....

Quelle instruction aurait-on pu écrire à la place de l'instruction " Pour *i* allant de 1 à 4 " ?

.....

Variables
p et i sont des nombres
Début
p prend la valeur 0
Pour *i* allant de 1 à 4
  p prend la valeur $3 \times p + 1$
Afficher *p*
Fin

Exercice 4

Jusqu'à présent, Tao ne recevait pas d'argent de poche. Comme au moins de janvier, il va fêter ses onze ans, ses parents lui font la proposition suivante : « On ne sait pas encore ce que l'on va te donner pour le mois de janvier. Mais à partir du mois de février, chaque mois, nous te donnerons la moitié de ce que nous te donnions le mois précédent plus 10 € » .

1° Dans cette question, les parents de Tao lui donnent 8 euros en janvier.

a) Combien d'argent de poche recevra-t-il en février ? ..... En mars ? ..... En juin ? .....

b) On a décrit, ci-dessous, la démarche qui permet de calculer le montant de l'argent de poche de Tao en juin à partir du montant de l'argent de poche de janvier.

On appelle M le montant de l'argent de poche de Tao (M change de valeur tous les mois).

Compléter les pointillés.

① Au départ M vaut ...
② Puis la *nouvelle valeur* de M s'obtient en .....
.....
③ On recommence à appliquer ② ..... fois
La dernière valeur de M est le montant de l'argent de poche de Tao au mois de juin.

2° Dans cette question, on ne connaît pas le montant de l'argent de poche de janvier.

a) Proposer un algorithme qui demande d'entrer l'argent de poche du mois de janvier et qui affiche l'argent de poche de Tao au mois de juin de la même année.

b) Ecrire cet algorithme sous *algobox* . Expliquer comment, à l'aide de la question 1°, on peut vérifier la cohérence de l'algorithme : .....
Effectuer cette vérification.

c) Proposer un deuxième algorithme qui demande d'entrer l'argent de poche du mois de janvier, qui affiche l'argent de poche de Tao au mois de juin de la même année et qui affiche aussi la somme totale que Tao aura reçu du mois de janvier au mois de juin.

Algorithmes (10) : Initiation à « Tant que ... faire ... »

Activité

Charlotte veut s'acheter une mini-chaîne HiFi : le modèle qui lui plaît coûte 99 €. Seulement, elle n'a pas d'économie car les 15 € d'argent de poche que lui donnent ses parents chaque semaine lui servent pour acheter des livres ou des vêtements ... Alors elle décide de mettre de côté une partie de son argent de poche : 1 euro la première semaine, 2 euros la deuxième semaine et ainsi de suite : chaque semaine elle économise un euro de plus que la semaine précédente et ceci tant que la somme économisée est insuffisante ...

1° Compléter le tableau suivant (en rajoutant autant de colonnes que nécessaire)

N : Numéro de la semaine	1	2	3	4	
S : Somme ajoutée à la tirelire					
M : Montant de la tirelire					

Au bout de combien de semaines pourra-t-elle s'offrir la mini-chaîne ? .....

2° a) On a décrit, ci-dessous, la démarche qui a permis de remplir le tableau de la question 1°.
Compléter les pointillés.

- Pour écrire cet algorithme, on a besoin de trois variables : N, S et M
- Au départ, on attribue la valeur 1 aux trois variables
- Puis : ① la nouvelle valeur de N s'obtient en ajoutant … à l'ancienne valeur de N
  ② la nouvelle valeur de S s'obtient en ajoutant … à l'ancienne valeur de S
  ③ la nouvelle valeur de M s'obtient en ajoutant … à l'ancienne valeur de M

  Et on recommence les instructions ① , ② et ③ tant que .....
- On obtient le nombre de semaines nécessaires pour économiser 99 € grâce à la dernière valeur de .....

b) A l'aide de la question a) compléter l'algorithme écrit ci-dessous pour qu'il sorte le nombre de semaines nécessaires pour économiser 99 €. (On admet que le nombre de semaines nécessaires est inférieur ou égal à 15 …)

```
Variables

Début
N prend la valeur .....


Tant que ..... , continuer à appliquer les instructions suivantes :
        ①
        ②
        ③
Sortir .....
Fin
```

c) Ecrire cet algorithme sous *algobox*. Vérifier que la réponse donnée par *algobox* est cohérente avec celle trouvée à la question 1° .

Exercice 1 Magalie, Robin et Lucie sont trois élèves qui s'entraînent à factoriser avec *Mathenpoche*.

Sachant que X désigne la note de l'élève à l'exercice qu'il vient de faire, compléter la dernière ligne du tableau suivant :

Magalie dit :	Robin dit :	Lucie dit :
Je continue à m'exercer sur le même type d'exercice jusqu'à ce que j'aie 10/10.	Je continue à m'exercer sur le même type d'exercice jusqu'à ce que ma note dépasse 7/10	Quand ma note à un exercice sera au moins 6/10, j'arrêterai.
Tant que X ... , Magalie continue à s'exercer.	Tant que X ... , Robin continue à s'exercer.	Tant que X ... , Lucie continue à s'exercer.

Exercice 2

On considère l'algorithme ci-contre :

1° Faire fonctionner « à la main » cet algorithme avec $n = 25$

Qu'affiche-t-il ? .....

2° Programmer l'algorithme sous *algobox* et vérifier la réponse précédente

Lire *n*
u prend la valeur *n*
Tant que $u > 7$, *u* prend la valeur $u - 7$
Afficher *u*

Exercice 3

On considère l'algorithme ci-contre où N est un nombre entier.

1° On lance l'algorithme et on entre N = 5.

Compléter le tableau suivant en ajoutant des lignes si nécessaire :

	Valeurs de :		
	N	I	X
Au départ			
Puis			

```
1     VARIABLES
2       X EST_DU_TYPE NOMBRE
3       I EST_DU_TYPE NOMBRE
4       N EST_DU_TYPE NOMBRE
5     DEBUT_ALGORITHME
6       LIRE N
7       I PREND_LA_VALEUR 1
8       X PREND_LA_VALEUR 1
9       TANT_QUE (I<=N) FAIRE
10        DEBUT_TANT_QUE
11        X PREND_LA_VALEUR 2*X
12        I PREND_LA_VALEUR I+1
13        FIN_TANT_QUE
14      AFFICHER X
15    FIN_ALGORITHME
```

2° Sans faire fonctionner l'algorithme, deviner ce qu'il afficherait si la valeur entrée était 7 : .....

3° L'algorithme n'affiche pas de réponse lorsque l'on supprime la ligne 12. Expliquer pourquoi.

.....

Exercice 4

On demande à des élèves d'écrire un algorithme qui respecte les conditions suivantes :

- l'algorithme demande à l'utilisateur d'entrer des nombres (autant de nombres que l'utilisateur le souhaite)
- l'algorithme continue de demander des nombres tant que l'utilisateur entre un nombre positif ou nul.
  L'utilisateur doit donc entrer un nombre négatif pour que l'algorithme arrête de demander des nombres
- l'algorithme calcule et affiche la somme des nombres entrés, sans compter le dernier.

1° On donne ci-dessous 4 algorithmes écrits par des élèves.

Parmi ces quatre algorithmes, trouver celui (ou ceux) qui respecte(nt) les conditions données.

Lorsqu'un algorithme ne convient pas, expliquer pourquoi.

Algorithme 1	Algorithme 2	Algorithme 3	Algorithme 4

S prend la valeur 0
Entrer X
Tant que X ≥ 0
  S prend la valeur S + X
Fin du tant que
Afficher S
	S prend la valeur 0
	Entrer X
	Tant que X ≥ 0
	  S prend la valeur S + X
	Fin du tant que
	Entrer X
	Afficher S
		S prend la valeur 0
		Entrer X
		Tant que X ≥ 0
		  S prend la valeur S + X
		  Entrer X
		Fin du tant que
		Afficher S
			S prend la valeur 0
			Entrer X
			Tant que X ≥ 0
			  Entrer X
			  S prend la valeur S + X
			Fin du tant que
			Afficher S

2° Un élève remarque qu'écrire un algorithme qui calcule la somme de nombres n'est pas très utile...

Alors, on change une condition afin que le but soit plus intéressant : au lieu d'afficher la somme des nombres entrés, l'algorithme doit afficher leur moyenne...

Modifier le bon algorithme pour qu'il affiche la moyenne des nombres entrés au lieu d'afficher leur somme.

En classe de seconde

Activités diverses

- Algorithmes : premiers pas
- Calculatrice et programmation : premiers pas avec la TI
- Fiche d'exercices pour introduire le « si-alors »
- La conjecture de Syracuse (*Si-alors / Boucles / Algobox)*
- Le car de supporter (*Fonction / si-alors*)
- Boucles « Pour » : entraînement
- Les variables de stockage (*logo*)
- Recherche de solution entière d'une inéquation *(Si-alors / Boucle pour /Algobox /Calculatrice)*
- Dichotomie (*Si-alors / Boucle pour*)

Algorithmes : premiers pas

1) Exemple : Suite d'opérations élémentaires

Un magicien demande à un spectateur : - de penser à un nombre entier; - de le remplacer par son double ; - de retrancher 3 au résultat obtenu ; - de multiplier le nouveau résultat par 6 ; - d'annoncer le résultat final obtenu.	1) Le spectateur pense au nombre 6, quel nombre annonce-t-il ? 2) même question s'il pense à 8. 3) Si le spectateur annonce 32 que dira le magicien ? 4) Le spectateur annonce 162 à quel nombre avait-il pensé ?

Point de vue du magicien

Donner une suite d'opérations élémentaires qui permettront au magicien de trouver le nombre choisi par le spectateur après que celui ci ait annoncé son résultat final.

2) Qu'est-ce qu'un algorithme ?

Un algorithme est un enchaînement d'étapes ou d'instructions à effectuer dans un certain ordre et dont la réalisation va permettre la résolution d'un problème donné.

Un algorithme doit être lisible par tous. On l'écrit en langage courant.
Un des intérêts est de pouvoir coder un algorithme dans un langage informatique afin qu'une machine (ordinateur, calculatrice...) puisse l'exécuter rapidement et efficacement.

3) Qu'est ce qui constitue un algorithme ?

Un début et une fin : Les algorithmes sont constitués d'un nombre fini d'étapes à exécuter dans un ordre bien défini, on peut donc considérer qu'il y a un début et une fin.

Des instructions : Durant l'enchaînement des étapes, ces étapes vont agir. On va les appeler des instructions.
Dans les instructions, on distingue 1- l'entrée des données
2- le traitement des données
3- la sortie des résultats

Des variables : Durant l'exécution d'un algorithme, on va avoir besoin de stocker des données, voire des résultats. Pour cela on utilise ce qu'on appelle des « variables ».
Une variable pourra être une chaîne de caractères, un nombre. Le contenu d'une variable peut être modifié au cours du déroulement de l'algorithme.

Ex 1 : On donne l'algorithme :

- Choisir deux nombres A et B
- Calculer A + 2B et remplacer B par le résultat obtenu
- Calculer B – A et remplacer A par le résultat obtenu
- Afficher A

a) Repérer la phase entrée, la phase traitement, la phase sortie.
b) Faire fonctionner cet algorithme avec A = 8 et B = 5, en remplissant le tableau suivant :

	Contenu de A	Contenu de B
Entrée		
Étape1 du traitement		
Étape 2 du traitement		
Sortie		

c) Essayer avec d'autres valeurs du couple (A, B). Avez -vous deviné ce que calcule cet algorithme ?

Ex 2 :On donne l'algorithme suivant

Entrée : donner une valeur à X
traitement : mettre X + 4 dans Y
mettre X×Y dans Y
sortie : afficher Y

environnement : X : variable numérique ; Y : variable numérique

1) Faîtes fonctionner cet algorithme pour X = 3 ; X = 5
(présenter votre réponse sous forme de tableaux)

2) Exprimez en fonction de X le nombre qui est affiché en sortie.

Ex 3 : On donne l'algorithme suivant

Entrée : donner une valeur à X
traitement : mettre 2X dans Y
mettre Y +3 dans Y
sortie : afficher Y

environnement : X : variable numérique ; Y : variable numérique

1) Faîtes fonctionner cet algorithme pour X = 3 ; X = 5

2) Exprimez en fonction de X le nombre qui est affiché en sortie.

Ex 4 : Écrire un algorithme qui permet d'échanger le contenu de deux variables X et Y

Ex 5 : Voici un algorithme :

Variables :

x, a , b , y quatre variables numériques

Entrée : Donner une valeur à x

Traitement :

Affecter à a la valeur x + 2

Affecter à b la valeur $\frac{1}{1+a^2}$

Affecter à y la valeur $a + \frac{1}{b}$

Sortie : Afficher y

1) Quel nombre sera affiché en sortie si on donne successivement 0 ; 1 et –2 en entrée ?

2) Donner l'expression algébrique de la fonction f qui à un nombre x donné en entrée associe le nombre y obtenu en sortie de l'algorithme.

Fiche algorithme : premiers pas

Ce qui a été fait antérieurement :

- Méthode de construction d'un nombre rationnel
- Fiche « Algorithme de Babylone ».
- Chaque fois que l'occasion s'est présentée, une présentation de calculs ou de démarche de résolution sous forme algorithmique (en décrivant les tâches à effectuer sous forme de listes).

Pré-requis : Règles de calculs vues au collège, pour le dernier exercice, notion de fonction.

Objectif : Mettre en place le vocabulaire : algorithme, entrée, traitement, sortie, variable.
Comprendre le principe de fonctionnement .
Améliorer la maîtrise des priorités à respecter dans des calculs.
progresser dans la notion de variable.

Durée : 40 minutes + 20 minutes

Déroulement :
Lors de la première séance :

- Les élèves font seuls pratiquement sans aide l'exemple 1).
- On lit ensemble, le 2) et le 3) les choses paraissent un peu artificielles présentées ainsi.
- Ex 1 : Les élèves travaillent en binôme. Pas de problème , la donnée du tableau les aide, pour le c) certains refont des tableaux pour présenter les résultats les autres (la majorité) fait les calculs de tête pour plusieurs exemples.
  Mais ce sont ceux qui ont fait des tableaux qui découvrent le plus rapidement ce que fait l'algorithme.
- Ex2 : Là encore une partie fait un tableau l'autre non, je demande à ceux qui n'ont pas fait de tableau et qui se sont trompés d'en faire un : l'élaboration du tableau n'est pas évidente ; par contre après, ils n'ont plus de problème.
- Ex3 : pas de problème
- l'heure touche à sa fin, je ne veux pas bâcler l'exo 4 , je demande aux plus rapides de commencer l'exo5 .

  Ex5 à finir pour la séance suivante.

Séance suivante :
Correction de l'exercice 5 : certains ont eu du mal , mais comprennent rapidement la correction.

Recherche de l'exo 4 : Là ils sont déroutés, je les laisse chercher.

Voici des exemples de recherche de deux élèves :

premier élève :
Il fait immédiatement un tableau et essaie avec des exemples

	x	y
entrée	5	10
x + y pour y	5	15
y-x pour x	10	15
y-x pour y	10	5
sortie	10	5

7	2
7	9
2	9
2	7

Il est plutôt content en discute avec son voisin qui malin, essaie avec des nombres négatifs.....et lui fait remarquer que cela ne marche pas.

Voilà ce qu'a fait le voisin,

choisir 2 nombres x et y
mettre x×y dans x
mettre x:y dans y
mettre x:y dans x

Puis voilà ce qu'ils ont fait ensemble:

	x	y
(dans y) x + y	x	x+ y
(dans x) y - x	y	x+ y
(dans y) y - x	y	x
	y	x

	x	y
(dans y) x×y	x	x×y
(dans x) y : x	y	y-x
(dans y) y : x	y	x
	y	x

	x	y
(dans y) y-x	x	y-x
(dans x) y + x	y	y-x
(dans y) y - x	y	-x
(dans x) x×-1	y	x

Franchement je trouve çà bien !!!!! Ils étaient contents, le signe des nombres leur a poser problème,

Dans plusieurs autres groupes cela fusait, et ceux qui ne trouvaient pas comme d'habitude du premier coup, commençaient à paniquer....
Une question est venue « est-ce qu'on peut ajouter une variable ? » Dommage que la question ait été posée tout fort car ceux qui voulaient continuer à chercher se sont sentis frustrés, on venait de leur donner l'idée !!!! C'est alors allé vite pour la moitié de la classe.

Plusieurs ont eu des difficultés à comprendre. J'ai proposé « Léo, Léa ». au tableau avec une petite phrase improvisée(voir autre fiche).... on a réfléchi un peu ensemble, je leur ai demandé d'essayer pour la prochaine séance..... Cela les a aider à comprendre, la notion de variable.

TD : Premiers pas pour faire un programme sur sa calculatrice

Objectif : Remplir le tableau de valeurs suivant : avec $f(x) = 2(x - 3)^2$

x	-5,7	-2,73	-2	-1,5	0	1,8	2,8	2,9	3	3,05	3,1	3,25	4,32
f(x)													

x ne varie pas à pas constant. Les menus « deftable » et « table », ne sont pas pratiques dans ce cas.

programmer sa calculatrice ;

Vous voulez : 1) Choisir une valeur pour la variable x
2) Effectuer le calcul $2(x - 3)^2$
3) Faire afficher le résultat.

Pour écrire un programme sur votre calculatrice avec TI82
Appuyer sur la touche « PRGM »
Choisir « Nouveau » (« NEW »)
Écrire un nom pour votre programme ici par exemple « FONC»
(Observez que le curseur clignote de façon différente : il est en mode écriture de lettres , vous trouvez les lettres au dessus des touches habituelles). Quand vous avez fini d'écrire le nom, appuyer sur la touche « ENTER »
Vous allez maintenant entrer ligne par ligne vos instructions, à chaque fin de ligne faites ENTER pour passer à la ligne suivante.

Entrée : saisie du nombre choisi	: INPUT X variante : PROMPT X	L'instruction input fera apparaître ? sur l'écran de votre calculatrice quand vous exécuterez votre programme. Le nombre que vous taperez alors, sera mis dans la case mémoire X de votre calculatrice vous trouverez INPUT en appuyant sur prgm, puis sur I/O. Cette instruction fait comme Input , mais fait apparaître X = ? à l'écran de la calculatrice lors de l'exécution. Vous la trouverez au même endroit qu'input.
Traitement : effectuer le calcul , le mettre dans la case mémoire Y	: $2(X - 3)^2 \rightarrow Y$	On met dans la case mémoire Y le résultat de $2(X - 3)^2$ vous trouverez → en faisant STO.
Sortie : afficher Y	: Disp Y	On demande d'afficher le contenu de Y Vous trouverez DISP en appuyant sur prgm, puis sur I/O.

Faire 2nd quit pour sortir du mode programmation

Maintenant vous allez faire fonctionner votre programme
Appuyer sur PRGM, choisir EXEC (exécuter),choisir le programme que vous voulez exécuter , faire « ENTER ».

Vous constatez que si vous voulez essayer plusieurs valeurs de X, vous êtes obligés de rappeler à chaque fois votre programme .

Pour éviter cela :
Retournez dans le menu PRGM, choisir EDIT, puis le nom du programme que vous voulez modifier (ici « FONC») ajouter la ligne :

: prgmFONC	Pour cela appuyez sur prgm, puis exec, puis choisissez FONC. Le programme se rappellera lui-même automatiquement. Quand vous serez en mode exécution et que vous voudrez arrêter, vous ferez 2nde quit

Autre idée : Fabriquer un programme qui peut servir pour des fonctions différentes.

On donne la fonction définie sur IR par $f(x) = x^2 + 1$.
a) Entrez l'expression de f(x) dans Y1.
b) Faire un programme de calcul qui permet de calculer les images d'un nombre par f en utilisant l'expression écrite dans Y1

La seule nouveauté est Y1 .

Recommencer :
nouveau programme, appeler ce nouveau programme FONC1 (pour se rappeler que l'expression sera dans Y1.

Puis, écrire les instructions suivantes : : Pompt X
: Y1(X)→Y appuyer sur Vars pour accéder à Yvars, choisir alors Y1
: DispY
: prgmFONC1

Tester votre programme.

Pour s'entraîner à faire des petits programmes sur sa calculatrice.

Ex 1 : *Calcul de volume*
1) Écrire un algorithme qui permet de calculer le volume d' un parallélépipède rectangle de côtés a, b, c.
2) Traduire cet algorithme sous forme de programme sur votre calculatrice.
3) Tester ce programme en prenant a = 1 ; b = 2, c = 3

Ex 2 : C*alcul de moyennes*
1) a) Écrire un algorithme, puis un programme qui permet de calculer la moyenne de deux notes x et y
b) Traduire cet algorithme en programme sur votre calculatrice.
2) Recommencer dans le cas où la note x est affectée du coefficient a et y du coefficient b.

Bilan de la fiche « calculatrice et programmation »

Pré-requis : Avoir déjà utilisé sa calculatrice, notion de fonction, on peut avoir déjà fait un peu d'algorithmique ou non.

Objectif : Traduire un algorithme dans un langage imposé (Ici TI, puis algobox)
Progresser dans la connaissance des possibilités de sa calculatrice.
Avoir un outil toujours à disposition pour tester un algorithme.

Durée, Conditions : 55 minutes en module (groupes de 17), presque tous ont une TI82. Les autres ont une Casio, il est demandé à ces derniers de se regrouper et d'essayer de trouver les menus adéquats eux mêmes.

Déroulement :

Je distribue la fiche, je procède de façon très directive jusqu'à l'écriture du nom du programme ; puis chacun va à son rythme, seul ou avec son voisin. Je dois intervenir plusieurs fois pour les aider à repérer les touches.

Ils remplissent le tableau donné avec une précision de 0,01 ; ceux qui ne trouvent pas comme les autres s'interrogent et cherchent leur erreur, souvent des étourderies.

Pour la deuxième idée, je leur demande, de **garder la même fonction** plutôt que de prendre celle écrite sur la fiche, l'idée n'était pas préméditée mais s'avère intéressante car ils testent avec les valeurs du tableau et s'aperçoivent tout de suite s'ils peuvent valider leur programme ou non. Là encore pas mal d'erreurs de frappe qu'ils détectent en général assez vite.

Pour l'exo 1 :
Réussi assez rapidement, là j'improvise et je demande à une élève de **traduire l'algorithme écrit sur algobox** ; (on a un ordinateur dans la classe et un vidéo projecteur) c'est une des élèves les plus en difficulté, mais elle réussit tout de suite.

Pour l'exo 2 :
Je dois d'abord reparler de moyennes pondérées, notion mal connue.
Là encore, j'envoie une autre élève faire le programme sur algobox, car c'est plus rapide à écrire que sur calculatrice.
On teste pour des valeurs simples calculées de tête au cours des explications préliminaires.
Ils doivent faire le programme sur leur calculatrice en travail maison.

Bilan :
Séance qui a bien fonctionné, les élèves ont beaucoup travaillé en autonomie. Ils ont vu l'intérêt de faire un programme et en demandaient d'autres.
Le fait que cinq élèves avaient une Casio n'a pas posé de problème, ils ont réussi à faire les programmes en s'aidant des indications données dans leur livre .

Algorithmes : « Si….Alors….Sinon »

Ex 1 :
Un magasin de photos propose le développement au tarif de 0,16 € l'unité ; le tarif est de 0,12 € pour une commande d'au moins 75 photos.

a) Que paiera-t-on pour le développement de 50 photos ? de 80 photos ?
b) Ecrire un algorithme qui demande le nombre de photos à développer et qui affiche le prix à payer.
c) Programmer cet algorithme sur Algobox. Tester avec les valeurs obtenues au a).

Ex 2 : Un magasin de reprographie propose un tarif dégressif :
Les 20 premières photocopies sont facturées à 10 centimes l'unité et les suivantes à 8 centimes l'unité.

a) Que paiera-t-on pour 15 photocopies ? pour 30 ?
b) Ecrire un algorithme qui demande à l'utilisateur le nombre de photocopies qu'il veut réaliser et qui affiche le prix qu'il devra payer.
c) Programmer cet algorithme sur Algobox, tester avec les valeurs obtenues au a).

Ex 3 :
Ecrire un algorithme qui demande deux nombres distincts (différents) et qui donne en sortie le plus grand des deux.
Amélioration : l'algorithme doit tester l'erreur de frappe qui consisterait à entrer deux nombres égaux.

Ex 4 :
Le plan est muni d'un repère.
Quatre points du plan A, B, C et D sont connus par leurs coordonnées.

1) Ecrire un algorithme qui permet de déterminer si le quadrilatère ABCD est un parallélogramme ou non.
2) Tester votre algorithme sur Algobox en prenant les cas suivants :

	Coordonnées de A	Coordonnées de B	Coordonnées de C	Coordonnées de D
Cas n°1	(-3 ;1)	(0 ; -2)	(3 ;0)	(0 ;3)
Cas n°2	(-3 ; 1)	(3 ; 0)	(0 ; -2)	(0 ; 3)
Cas n°3	(1 ; 2,5)	(-1 ; -1)	(0 ; -2)	(2 ; 1,5)
Cas n°4	(-0.9 ; 1.2)	$(\frac{5}{3}$; -2.1)	(4 ; 2)	(1,5 ; 5,3)

3) Améliorer le programme en demandant le tracé du quadrilatère ABCD (choisir : dessiner dans un repère)

Programmation sur Calculatrice, d'une structure alternative **« Si….Alors….Sinon »**

Avec l'exemple de l'exo 1 :

Algorithme	Programme TI	Programme Casio
Entrée : saisir le nombre de photos N Traitement : Si N < 75 alors afficher 0,16 × N Sinon afficher 0,12× N Fin Si Sortie : faite en cours de traitement	PROGRAM : photos : Prompt N : If N < 75 : Then : Disp 0,16 × N : Else : Disp 0,12× N : End	Photos ?→N If N < 75 Then 0,16 × N Else 0,12 × N Ifend

Entraînement : Traduire dans le langage de votre calculatrice les algorithmes écrits aux exercices 2 ; 3 et 4

Commentaires de la fiche « si…….alors…….sinon ».

Pré-requis :
Algos : Entrées - sorties. Avoir déjà pratiqué le logiciel Algobox et la programmation sur calculatrice.
Maths : Coordonnées d'un point dans le plan muni d'un repère, coordonnées du milieu d'un segment.

Objectif : Mettre en place la notion de structure alternative.
Faire de petits algorithmes et les faire fonctionner avec « Algobox », la calculatrice.

Durée : Sur 2 séances : 40 minutes (module) + 30 minutes (classe entière).

Déroulement : En Module : travail en binôme

Exercice 1 :
Question a) : Pas de problème.

Question b) : Les élèves ne savent pas trop comment s'y prendre pour la forme contrainte d'écriture de l'algorithme alors qu'ils ont bien compris le problème posé.
On rappelle les trois grandes lignes : entrée - traitement sortie.
On complète ensemble les étapes entrée-sortie, ils se débrouillent seuls pour le traitement.

La majorité utilise deux fois le « si » et écrivent :
Si le nombre de photos est < 75 alors afficher le nombre de photos fois 0,16.
Si le nombre de photos est ≥ 75 alors afficher le nombre de photos fois 0,12.
Le « sinon » n'est pas naturel.

Question c) : Les élèves ont déjà programmé sur algobox, mais ils font encore beaucoup d'erreurs.
La déclaration des variables les bloque, je leur conseille d'essayer d'écrire le traitement sans s'occuper de ces déclarations, la moitié comprend vite.
Pour le nom des variables plusieurs écrivent « nbrephoto » pour le nombre de photos ou « prix » pour le prix à payer, choisir juste une lettre « n » ou « p » n'est pas naturel.
Ils arrivent à faire un programme qui marche, après les classiques oublis : « lire.. » et « afficher … » mais très peu ont utilisé le « sinon ».
Au final avec de l'aide tout le monde réussit, l'exercice, on corrige en utilisant le sinon.

Exercice 2 :
Question a) : Quelques erreurs pour le 30.
Question b) : La plupart saute cette question pour se précipiter sur l'écrire du programme sur Algobox.
Je les laisse faire, mais constate que seul une dizaine réussit un programme qui marche.
Pour les autres, je leur impose de revenir en arrière….refaire a) et faire b) Ils le font contraints et forcés !!!! Mais le fait d'écrire sur le papier ce qu'ils veulent faire les débloque. C'est le problème mathématique qui les gênait.

Retour à la programmation : Toujours le problème des variables !!!!!
Par contre cette fois le sinon est presque toujours utilisé.
Tous arrivent à un programme qui tourne même s'il n'est pas optimisé au niveau du choix des variables.

Exercice 3 :
La première partie de l'exercice est traitée assez rapidement, par ceux qui ont eu le temps de la commencer (la moitié des élèves).
L'amélioration demandée provoque de nombreuses discussions : où placer le 1ier test ? Cela prend du temps.

Pour la séance suivante je demande de faire la question 1) de l'exo 4, en facultatif la question 2) et je demande à ceux qui n'ont pas commencer l'ex 3 de le faire sans l'amélioration.

• Séance suivante :
15 minutes pour corriger l'exo 3 (sans l'amélioration: la moitié l'avait bien faite en TD, ce n'est pas prioritaire pour les autres) et le début de l'exo 4. La programmation sur Algobox est demandée en travail personnel.
Puis je prends 10 minutes pour faire sur TI 82 le programme de l'ex1.

Bilan :
La notion de variable est difficile, un travail préalable sur cette notion (voir fiche…) aurait été utile. Le sinon n'est pas naturel, mais s'acquiert bien après deux ou trois exemples.
Il vaut mieux pratiquer moins longtemps, mais plus régulièrement que je ne l'avais fait sur le logiciel Algobox, sinon ils oublient vite car les notions ne sont pas encore bien acquises.

Conjecture de Syracuse

1) De quoi s'agit-il ?

Algorithme de Collatz (Mathématicien allemand 1910-1990):

1) Choisir un nombre entier.
2) • Si le nombre est pair, on le divise par 2 et on obtient un nouveau nombre.

   • Si le nombre est impair, on le multiplie par 3, on ajoute 1 au résultat et on obtient un nouveau nombre.

3) On recommence la procédure décrite au 2) avec le nouveau nombre obtenu.

On obtient une suite d'entiers positifs.

Ecrire la suite obtenue si le nombre choisi au départ est 14. Que constatez-vous ?

Conjecture de Syracuse ou encore : conjecture de Collatz

(*Rappel :* une conjecture est une supposition, une affirmation qui n'est pas démontrée)

Quel que soit le nombre entier non nul choisi au départ, on finit toujours par obtenir 1 dans la suite obtenue avec l'algorithme de Collatz

La suite des nombres obtenue est appelée **le vol** du nombre de départ,
les nombres de la suite sont appelés **les étapes** du vol,
le plus grand nombre obtenu dans la suite est appelé **l'altitude maximale** du vol,
et le nombre d'étapes avant d'obtenir 1 est appelé **la durée** du vol.

Vous pouvez faire un autre essai avec un nombre de votre choix. (Attention selon le nombre choisi au départ la durée du vol peut être longue !!!!)

Remarques : Actuellement cette conjecture n'est pas démontrée, et aucun contre exemple n'a été trouvé.
Pourquoi Syracuse ? La conjecture émise par Collatz en 1937, fut diffusée autour de 1960 à l'université de Syracuse aux USA.

2) Ecriture de l'algorithme

a) Ecrire un algorithme que vous transcrirez sur algobox avec pour condition d'arrêt l'obtention du chiffre 1.
b) Modifier votre programme pour qu'il affiche « les étapes du vol ».
c) Incorporer à ce programme un compteur qui permettra de déterminer le nombre d'étapes avant l'obtention éventuelle du 1.
d) Améliorer votre programme pour qu'il affiche l'altitude du vol.

Commentaires de la fiche « conjecture de Syracuse »

Pré requis : Notion de « si … alors …. sinon »
Notion d'utilisation de « algobox ».

Objectif : Réinvestissement du « si alors sinon ».
Introduire les boucles avec conditions d'arrêt : « tant que ».
Introduire un compteur.

Durée : 1 séance en module

Déroulement :

• Travail en autonomie par binôme.

• Question 1) Au début ils sont un peu surpris de voir les nombres augmenter et se disent qu'ils vont y passer l'heure, les plus rapides rassurent les autres. Ils choisissent en général un autre exemple et s'aperçoivent qu'ils peuvent réutiliser les calculs déjà faits au bout d'un certain stade.

• 2) a) Ils se lancent en général directement sur Algobox, rapidement ils ont des difficultés avec la boucle et le critère d'arrêt. J'explique pour tout le monde au tableau

Pour l'écriture de la condition d'arrêt, certains ont mal compris et l'écrivent en Français…

Autre difficulté : traduire « n pair »

Comme aucun affichage n'est demandé, ils sont un peu déroutés, j'insiste pour qu'ils aillent voir dans le mode pas à pas là, ils peuvent voir ce qui se passe et ils retrouvent les nombres qu'ils avaient obtenus manuellement.

• 2) b) Ils ont trouvé dans l'ensemble une méthode.

• 2) c) Très peu (5-6) ont eu le temps d'aborder cette question.

Exemple d'algorithme pour le 2) a)

```
1   VARIABLES
2     N EST_DU_TYPE NOMBRE
3   DEBUT_ALGORITHME
4     LIRE N
5     TANT_QUE (N!=1) FAIRE
6       DEBUT_TANT_QUE
7       SI (N%2==0) ALORS
8         DEBUT_SI
9         N PREND_LA_VALEUR N/2
10        FIN_SI
11        SINON
12          DEBUT_SINON
13          N PREND_LA_VALEUR 3*N+1
14          FIN_SINON
15      FIN_TANT_QUE
16  FIN_ALGORITHME
```

Exemple d'algorithme pour le 2) b) et c)

```
1   VARIABLES
2     N EST_DU_TYPE NOMBRE
3     D EST_DU_TYPE NOMBRE
4   DEBUT_ALGORITHME
5     LIRE N
6     D PREND_LA_VALEUR 0
7     TANT_QUE (N!=1) FAIRE
8       DEBUT_TANT_QUE
9       SI (N%2==0) ALORS
10        DEBUT_SI
11        N PREND_LA_VALEUR N/2
12        D PREND_LA_VALEUR D + 1
13        AFFICHER N
14        AFFICHER " ; "
15        FIN_SI
16        SINON
17          DEBUT_SINON
18          N PREND_LA_VALEUR 3*N+1
19          D PREND_LA_VALEUR D+1
20          AFFICHER N
21          AFFICHER " ; "
22          FIN_SINON
23      FIN_TANT_QUE
24    AFFICHER "la durée du vol est : "
25    AFFICHER D
26  FIN_ALGORITHME
```

Problème des cars de supporters

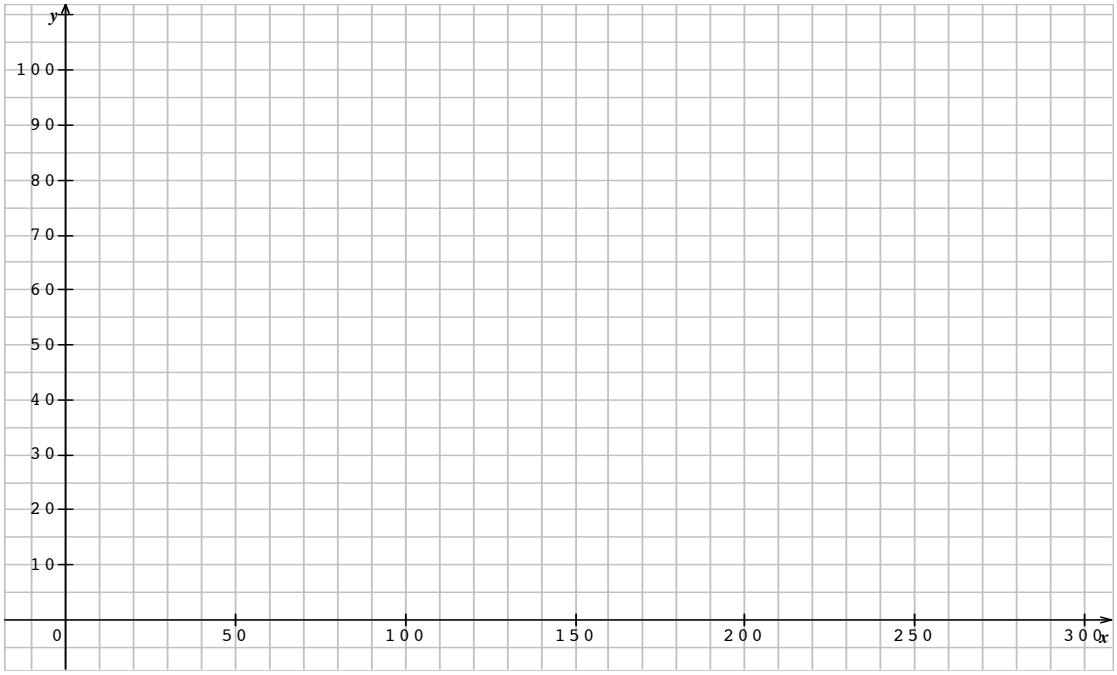
Une entreprise de transport possède 4 cars de 50 places chacun et se propose d'assurer le transport des supporters d'une équipe de rugby. Chaque car se loue 800 € tout compris.

Partie A :

1. Quel est le nombre maximal de supporters que cette entreprise peut transporter ?
2. On suppose que 50 supporters se rendent au stade ; calculez le prix à payer par supporter au centime près.
3. On suppose que 160 supporters se rendent au stade ; calculez le prix à payer par supporter au centime près.
4. Compléter le tableau suivant :

Nombre de supporters	10	20	40	50	60	80	100	120	150	160	200
Prix par supporter											

5. On souhaite représenter graphiquement le prix par supporter en fonction du nombre de supporters se rendant au stade ; placer les 11 points obtenus dans la question 4.



Partie B :

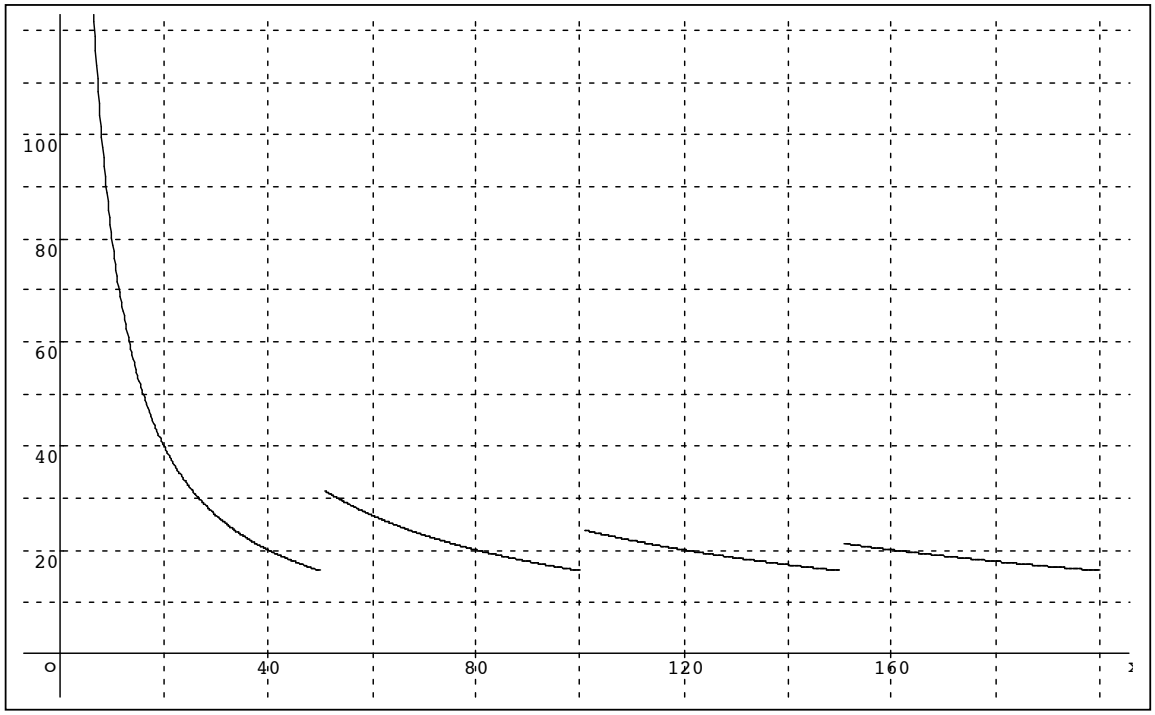
1. Ecrire un algorithme qui permet d'obtenir de façon automatique le prix à payer par supporter en fonction du nombre de supporters se rendant au stade. Testez votre algorithme sur Algobox.

2. Pour aller plus loin :

Représenter graphiquement (toujours en utilisant Algobox) le prix à payer par supporter en fonction du nombre de supporters se rendant au stade.

Correction

Nombre de supporters	10	20	40	50	60	80	100	110	150	160	200
Prix par supporter	80	40	20	16	26,67	20	16	21,81	16	20	16



Commentaires fiche cars de supporters.

Pré requis : Algorithme : si alors sinon, boucles pour.
Maths : fonction, représentation graphique.

Objectif : Réinvestir les notions d'algorithmiques déjà abordées.
Fonction homographique par intervalles.

Durée : 1séance en module.

Déroulement :
Partie A : Des difficultés au niveau de la représentation graphique, j'ai laissé afin qu'ils constatent leurs erreurs grâce au logiciel Algobox.

La partie B : J'ai dû beaucoup intervenir pour les aider, mais la plupart sont arrivés faire les deux questions, et ont été convaincus de l'intérêt de la programmation.

Voici des exemples de solutions pour les questions 1 et 2

Nombre de cars nécessaires et prix à payer en fonction du nombre de supporters N (compris entre 1 et 200)

```
1   VARIABLES
2     p EST_DU_TYPE NOMBRE
3     nb EST_DU_TYPE NOMBRE
4     N EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     AFFICHER "nombre de supporters ?"
7     LIRE N
8     SI (N/50==floor(N/50)) ALORS
9       DEBUT_SI
10      nb PREND_LA_VALEUR N/50
11      FIN_SI
12      SINON
13        DEBUT_SINON
14        nb PREND_LA_VALEUR floor(N/50)+1
15        FIN_SINON
16    p PREND_LA_VALEUR nb*800/N
17    AFFICHER "le nombre de cars de nécessaire est: "
18    AFFICHER nb
19    AFFICHER "le prix à payer par supporter est : "
20    AFFICHER p
21    AFFICHER "   euros"
22  FIN_ALGORITHME
```

Représentation graphique de la fonction qui donne le prix à payer en fonction du nombre de supporters (compris entre 1 et 200)

```
1   VARIABLES
2     I EST_DU_TYPE NOMBRE
3     p EST_DU_TYPE NOMBRE
4     nb EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     POUR I ALLANT_DE 1 A 200
7       DEBUT_POUR
8       SI (I/50==floor(I/50)) ALORS
9         DEBUT_SI
10        p PREND_LA_VALEUR 800/50
11        TRACER_POINT (I,p)
12        FIN_SI
13        SINON
14          DEBUT_SINON
15          p PREND_LA_VALEUR (floor(I/50)+1)*800/I
16          TRACER_POINT (I,p)
17          FIN_SINON
18      FIN_POUR
19  FIN_ALGORITHME
```

Boucles « Pour » : entraînement

Ex 1 :

Voici un algorithme :
Variables
N et S : nombres
Initialisation
- Donner à S la valeur 0

Traitement
- Choisir un nombre entier N
- Pour i allant de 1 à N avec un pas de 1 faire :
  Donner à S la valeur S + i^2
- Fin du Pour

Sortie :
- Afficher S

a) Tester cet algorithme en prenant N = 4. Pour cela remplissez le tableau suivant :

	N	S	i
initialisation		0	
traitement	4	0	
	4	0	1
Sortie			

b) Que calcule cet algorithme ?
c) Programmer cet algorithme sur votre calculatrice.
d) Reprendre votre programme, et choisir un pas de 0,5. Qu'avez-vous calculé ?
e) Programmer cet algorithme avec algobox ; pouvez vous changer le pas ?

Ex 2 : Remarque si la valeur du pas n'est pas indiqué, on prend 1 pour ce pas.

Voici un algorithme :

Variables
N et P : nombres

Initialisation
- Donner à P la valeur 1000

Traitement
- Choisir un nombre entier N
- Pour i allant de 1 à N faire :
  Donner à P la valeur $\frac{1}{2}$ P + i
- Fin du Pour

Sortie :
- Afficher P

Tester cet algorithme en prenant N = 4.
Pour cela remplissez le tableau suivant :

	N	P	i
initialisation		1000	
traitement	4	1000	
	4	1000	
	4	1000	1
Sortie			

Ex 3 :

Voici un algorithme :

Variables
N et P : nombres

Initialisation
- Donner à P la valeur 1000

Traitement
- Choisir un nombre entier N
- Pour i allant de 1 à N faire :
  Donner à P la valeur $\frac{1}{2}$ P + 20i
- Fin du Pour

Sortie :
- Afficher P

Tester cet algorithme en prenant N = 4.
Pour cela remplissez le tableau suivant :

	N	P	i
initialisation		1000	
traitement	4	1000	
	4	1000	1
Sortie			

Ex 4 :

Voici un algorithme :

Variables
N , P et S : nombres

Initialisation
- Donner à S la valeur 0

Traitement
- Choisir un nombre P
- Choisir un nombre entier N
- Pour i allant de 1 à N faire :
  Donner à P la valeur $\frac{1}{2}$ P + 20
  Donner à S la valeur S + P
- Fin du Pour

Sortie :
- Afficher le couple (N , P)
- Afficher le couple (N , S

a) Tester cet algorithme en prenant P = 10 000 et N = 4.
Pour cela remplissez le tableau suivant :

	N	P	S	i
initialisation				
traitement				
				1
Sortie				

Ex 5 : Choisir : boucle « Pour » ou boucle « tant que »
Une machine outil perd 10% de sa valeur chaque année.
1) a) Ecrire un algorithme qui permet de calculer sa valeur cinq ans après son achat. Le programmer sur votre calculatrice.
b) Tester votre algorithme en prenant comme valeur de départ 10 000 €.
2) Ecrire un autre algorithme qui affiche le nombre d'années nécessaires pour que la valeur soit divisée par deux. Le programmer sur votre calculatrice.

TP

Les variables de stockage

Seconde 4

Partie A – Principe

Votre calculatrice vous permet de mémoriser des résultats de calcul dans des variables A, B, C, etc. Tous les langages informatiques offrent cette possibilité. Le langage LOGO aussi bien entendu.

En LOGO l'instruction « **soit** "température 25 » a pour effet de stocker le nombre 25 dans la variable température.

- Le nom d'une variable n'est pas limité au 26 lettres de l'alphabet, n'importe quel mot peut convenir.
- Le mot est logiquement précédé d'un double guillemet pour que la machine LOGO comprenne bien qu'il s'agit d'un mot et non d'une primitive.
- Cette instruction ne peut pas être utilisée sur la ligne de commande mais uniquement à l'intérieur d'une procédure.

Il est essentiel de se représenter une variable comme un genre de tiroir :

- le mot température est écrit sur l'étiquette du tiroir ;
- le nombre 25 est rangé à l'intérieur du tiroir.

Si vous exécutez maintenant l'instruction « **soit** "température 32 », la machine LOGO ouvre le tiroir ayant l'étiquette température, retire son contenu (c'est-à-dire retire le nombre 25) et place à l'intérieur le nombre 32.

1.

```
1: pour proc
2:     soit "température 25
3:     ecris "température
4: fin
```

Quel effet va avoir l'exécution de cette procédure ?

Créez la procédure **proc** et exécutez-la. Selon vous, le comportement était-il prévisible ?

La primitive **chose** permet de lire le contenu de la variable. Ainsi, l'instruction « **chose** "température » retourne le contenu de la variable, c'est-à-dire 25.

2. Corrigez la procédure **proc** pour qu'elle écrive la valeur de la température dans la fenêtre de texte.

Lire le contenu d'une variable est une action qui revient sans arrêt dans un programme informatique.

Pour nous faciliter la vie, la machine LOGO nous permet d'écrire « :température » à la place de « **chose** "température ». Remarquez bien le symbole « : » au début du mot.

3. Modifiez à nouveau la procédure **proc** en appliquant cette écriture abrégée.

Les variables peuvent être utilisées pour stocker n'importe quel type d'objet LOGO, c'est-à-dire aussi bien des nombres, des mots ou des listes.

Partie B – Somme de deux fractions

1. Effectuez à la main les sommes suivantes en détaillant les calculs :

(a)

$$\frac{2}{3}+\frac{7}{5}=$$

(b)

$$\frac{11}{19}+\frac{3}{10}=$$

2. Concevez et mettez au point une procédure « **sommefraction** :frac1 :frac2 » qui retourne la somme des deux fractions :frac1 et :frac2.

Les deux arguments :frac1 et :frac2 sont des listes de deux nombres (par exemple, « [11 19] » désigne la fraction $\frac{11}{19}$).

La fraction somme est elle même retournée sous la forme d'une liste.

Partie C – Comparaison de deux procédures

```
1: FONCTION fonc1(X)
2: Y prend la valeur X^2
3: Y prend la valeur Y + 3 × X
4: Y prend la valeur Y + 1
5: Y prend la valeur Y^2
6: retourner Y
```

```
1: FONCTION fonc2(X)
2: Y prend la valeur 1
3: C prend la valeur X
4:   Répéter 4 fois
5:     Y prend la valeur Y × C
6:     C prend la valeur C + 1
7:   Fin du répéter
8: Y prend la valeur Y + 1
9: retourner Y
```

1. **(a)** Calculez à la main la valeur retournée par la commande « **fonc1** 1 ». Détaillez les calculs :

(b) Calculez de même la valeur retournée par la commande « **fonc1** 2 ».

2. **(a)** Calculez à la main la valeur retournée par la commande « **fonc2** 1 ». Détaillez les calculs :

(b) Calculez de même la valeur retournée par la commande « **fonc2** 2 ».

3. Traduire les deux procédures ci-dessus en langage LOGO. Testez les puis rédigez les ci-dessous.

4. Quelle conjecture pouvez-vous faire concernant ces deux procédures ?

5. **(a)** Proposez une expression algébrique qui calcule **fonc1**(X).

(b) Proposez une expression algébrique qui calcule **fonc2**(X).

6. Prouvez que les deux expressions algébriques sont équivalentes.

Soit $f(x) = x(x+8)$. On s'intéresse à l'inéquation (I) : $f(x) \geq 120$.

Partie 1 : Savoir si un nombre est solution ou non

1) 10 est-il solution de (I) ? -5 est-il solution ?

2) Ecrire un algorithme qui à partir d'un nombre x, affiche s'il est ou non solution de (I).

3) Programmer cet algorithme avec Algobox et tester le avec différentes valeurs : 10 ; -5 ; 7,5…..

Partie 2 : Recherche du plus petit entier positif n solution de (I)

1) En utilisant le programme précédent, compléter le tableau ci-dessous :

n	0	1	2						
$f(n) \geq 120$ Oui ou Non									

Quel est le petit entier positif n solution de (I) ?

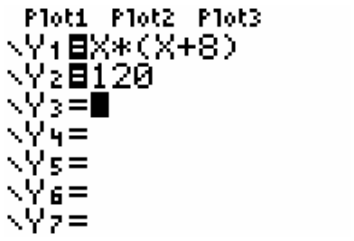
2) Voici un algorithme qui affiche directement cette solution :

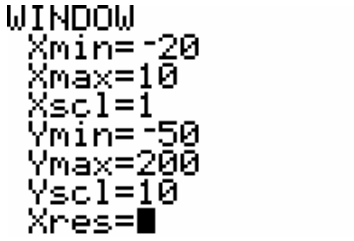
```
1     VARIABLES
2       n EST_DU_TYPE NOMBRE
3     DEBUT_ALGORITHME
4       n PREND_LA_VALEUR 0
5       TANT_QUE (F1(n)<120) FAIRE
6         DEBUT_TANT_QUE
7         n PREND_LA_VALEUR n+1
8         FIN_TANT_QUE
9       AFFICHER n
10    FIN_ALGORITHME
11
12   Fonction numérique utilisée :
13   F1(x)=x*(x+8)
```

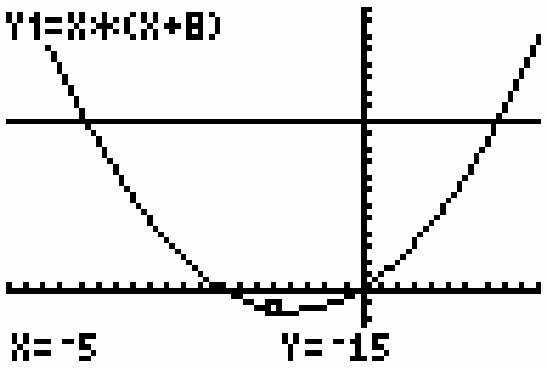
Programmer cet algorithme avec Algobox et tester-le en mode pas à pas.

Partie 3 : Interprétation graphique

1) Utiliser votre calculatrice pour obtenir les écrans suivants :







2) D'après la courbe représentant f, -5 est-il solution de (I) ?

3) Retrouver graphiquement le petit entier positif n solution de (I).

Partie 4 : Coût d'un algorithme

Reprenons l'algorithme de la partie 2 :

1) Montrer que le calcul de $f(x)$ comporte deux opérations de calculs. En déduire qu'une étape de traitement (ligne 5 à 8) comporte trois opérations de calculs.

2) Déterminer alors le nombre total de calculs nécessaires.

3) On s'intéresse maintenant à chercher le petit entier positif n solution de : $f(x) \geq 1200$.
Modifier l'algorithme de la partie 2 pour répondre à la question posée.

Quel est le nombre de calculs effectués par cet algorithme ?

4) On veut réduire le coût en réduisant le nombre de calculs effectués, on programme alors cet algorithme :

```
1     VARIABLES
2       n EST_DU_TYPE NOMBRE
3     DEBUT_ALGORITHME
4       n PREND_LA_VALEUR 0
5       TANT_QUE (F1(n)<1200) FAIRE
6         DEBUT_TANT_QUE
7         n PREND_LA_VALEUR n+10
8         FIN_TANT_QUE
9       n PREND_LA_VALEUR n-10
10      TANT_QUE (F1(n)<1200) FAIRE
11        DEBUT_TANT_QUE
12        n PREND_LA_VALEUR n+1
13        FIN_TANT_QUE
14      AFFICHER n
15    FIN_ALGORITHME
16
17  Fonction numérique utilisée :
18  F1(x)=x*(x+8)
```

Compléter le tableau suivant, la première ligne donne les valeurs prises par n dans l'algorithme :

n	0	10	20				
$f(n) \geq 120$ Oui ou non							

Pourquoi cet algorithme est-il plus performant ?

Quels est ici le nombre de calculs nécessaires pour répondre à la question ?

Info : La notion de coût d'un algorithme est importante. Pour calculer ce coût, on compte le nombre d'opérations mais aussi le nombre d'affectations, de tests effectués…..on cherche à minimiser le coût pour minimiser le temps de calcul.

Couper en deux, encore et encore : la dichotomie

I : Jeu du nombre inconnu

Un élève volontaire choisit un nombre entier compris entre 0 et 256.
Un autre élève cherche à deviner ce nombre, en adoptant la stratégie suivante :

Etape 1 : Il propose le nombre « milieu » 128 . L'élève qui a choisi le nombre dit " c'est plus " ; "c'est moins " ou "c'est juste".
Il note au tableau : le numéro de l'étape, le nombre proposé, l'intervalle dans lequel se trouve le nombre à trouver et son amplitude

Etape 2 : Il propose le « milieu » du nouvel intervalle obtenu.

Il recommence ainsi jusqu'à obtenir le nombre cherché.

1) Que peut-on dire des amplitudes des intervalles d'une étape à la suivante ?
2) Pourquoi est-on certain de trouver le bon nombre, quel sera le nombre maximum d'étapes par cette méthode ?

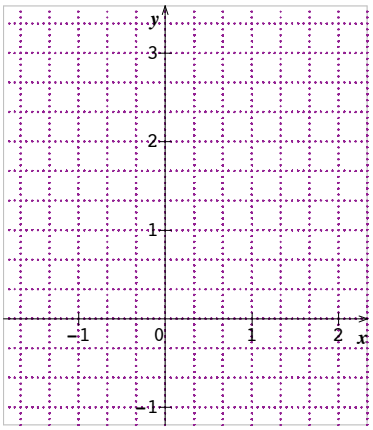
Précisions de vocabulaire :

Soit [a; b] un intervalle, le nombre b – a est appelée « amplitude » de l'intervalle.

Le nombre $\frac{(a+b)}{2}$ est le « centre » de l'intervalle

II : Résolution approchée d'une équation

« On cherche à résoudre dans $\mathbb{R}$ l'équation $x^2 = x + 1$. »



Dans un premier temps on va chercher une valeur approchée des solutions si elles existent.

1°) Graphiquement
Tracer, dans le repère ci-joint, la courbe C d'équation $y = x^2$,
puis la droite D d'équation $y = x + 1$.
a) Conjecturez le nombre de solutions de l'équation donnée.

b) Donnez un encadrement par deux entiers consécutifs de ces deux solutions éventuelles.

2°) A l'aide de la calculatrice et d'un algorithme
Résoudre l'équation $x^2 = x + 1$ équivaut à résoudre $x^2 – x – 1 = 0$,
on va chercher une valeur approchée de la solution positive éventuelle de l'équation $x^2 – x – 1 = 0$.

On note f la fonction définie sur $\mathbb{R}$ par $f(x) = x^2 – x – 1$.

1) Localisation de la solution éventuelle
a) Avec la calculatrice, conjecturer le tableau de variation de f sur $\mathbb{R}$. On admet que cette conjecture est vraie.

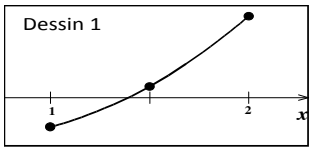
b) Calculez f(1), puis f(2) , pourquoi peut-on affirmer que « si l'équation f(x) = 0 possède une solution positive alors cette solution appartient à [1 ; 2] ».
On admet que cette solution existe.

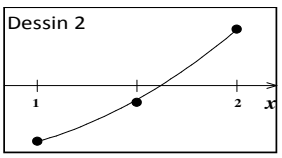
2) Dichotomie à la main

a) Etape 1 : • On calcule le centre de l'intervalle [1 ; 2], c'est : $\frac{(1+2)}{2} = 1{,}5$.

• Calculer f(1,5) à la calculatrice, est-il plus petit ou plus grand que 0 ?

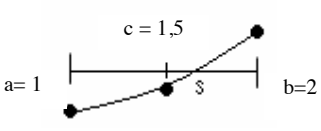
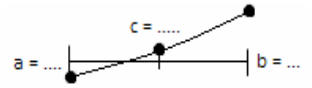
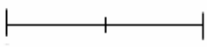
• Voici deux dessins. .
L'un des deux représente la fonction f sur [1 ; 2].
Lequel ?
Justifier : .....





• Dans quel intervalle se trouve la solution cherchée s : [1 ; 1,5] ou [1,5 ; 2] ?
ainsi, on peut recommencer ce qui précède en prenant a = et b =

b) On continue de la même façon :
Compte tenu de l'observation faite à l'étape 1 , on peut recommencer ce qui a été fait à cette étape en prenant cette fois : a = et b =
Compléter ce tableau pour les 6 premières étapes :

Etape n°	Centre c	L'image du centre est : « > 0 » ou « < 0 »?	illustration	La solution **s** appartient à [a ; b] avec	Amplitude de l'intervalle
début				a = 1 b = 2	
1	1,5	f(1,5) < 0		a = b =	
2					
3					
4					
5					
6					

c) En combien d'étapes a-t-on une valeur approchée à 0,1 près de la solution ?
Combien faut-il d'étapes pour avoir la solution à 0,01 près ?

3) Un algorithme pour programmer : Ici plusieurs approches sont possibles, voir autres idées sur fiche annexe (1)

Objectif : écrire un algorithme qui affiche l'intervalle obtenu après un nombre suffisant d'étapes pour que la longueur de cet intervalle soit inférieure à une longueur L donnée.

a) Ci-dessous, compléter la démarche qui a permis de remplir le tableau de la question 2°)**2)b)**.

Les nombres a et b sont les bornes de l'intervalle auquel appartient **s.** Au départ, a = …. et b = ……..

Le nombre e est l'écart entre a et b . Au départ, e = …..

Les 3 étapes suivantes ont pour but de réduire l'amplitude L de l'intervalle [a ;b]

① Le nombre c est le centre de l'intervalle [a ;b], il se calcule de la façon suivante : .....

② On calcule l'image de …… par *f*

Si ..... , alors .....

Si ..... , alors .....

③ On donne à e la valeur .....

Et on recommence les étapes ① , ② et ③ tant que .....

Les dernières valeurs de a et de b sont les bornes d'un intervalle qui contient s et dont l'amplitude est inférieure ou égale à L.

b) A l'aide de la question précédente, proposer un algorithme qui affiche les bornes a et b d'un intervalle d'amplitude inférieure ou égal à L, qui contient s, **avec les contraintes suivantes :**

1) Vous utiliserez uniquement des instructions prises parmi les instructions suivantes.

- Certaines « briques » ne serviront pas, d'autres seront utilisées plusieurs fois.

Lire la variable……
Affecter la valeur……..à la variable……..
Afficher……..
Si……..alors…….(éventuellement sinon…..)
Fin Si
Pour …..allant de…….à ……..
Fin Pour
Tant que…….faire……..
Fin tant que

2) Vous préciserez ce que signifie chacune des variables que vous aurez utilisées

c) Programmer cet algorithme sur votre calculatrice (ou sur algobox) et testez votre programme avec la fonction f définie à la question 2°) en prenant : a= 1; b= 2 et L = 0,1.

d) Donnez une valeur approchée de la solution cherchée à 0,01 près.

III Valeur exacte de la solution cherchée : le nombre d'or :

1) Prouvez que pour tout réel x : $x^2 - x - 1 = (x - \frac{1}{2})^2 - \frac{5}{4}$
2) En déduire que l'équation f(x) = 0 possède deux solutions dans IR. La solution positive est appelée « Nombre d'or » et souvent noté φ.
3) Bizarre, bizarre :

1) Prouvez que : $\varphi = 1 + \frac{1}{\varphi}$.

2) Calculez : $1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\varphi}}}$

Annexe (1) : Variantes pour la partie : un algorithme pour programmer.

Variante 1 : Après le II)2°)2)b), un algorithme àcompléter est demandé directement aux élèves,

3) Un algorithme

On veut écrire un algorithme qui affiche l'intervalle obtenu après un nombre suffisant d'étapes pour que la longueur de cet intervalle soit inférieure à une longueur L donnée.

a) Compléter l'algorithme suivant :

```
Variables
    a, b, L, m
entrée
    saisir a, b ( avec a < b), L

traitement
    I TANT QUE b – a > .....
    I m prend la valeur .....
      I si f(m) > 0
                  alors a prend la valeur ....
                  sinon .....
       I FIN SI
    I FIN TANT QUE
sortie
   afficher .....
```

b) Programmez cet algorithme sur votre calculatrice et testez votre programme avec la fonction f, a= 1; b= 2 et l = 0,1.

c) Donnez une valeur approchée de la solution cherchée à 0,01 près.

Variante 2 : après le II)2°)2)b), avant de demander aux élèves un algorithme, travail sur tableur.

3) Retrouvez le tableau précédent à l'aide d'un tableur.

	A	B	C	D	E
1	Calcul de la valeur approchée du nombre d'or par dichotomie				
2					
3	f(x)=x²-x-1				
4					
5	a	b	m	f(m)	b-a
6	1	2			
7					
8					
9					
10					
11					
12					

Les formules à écrire et à recopier vers le bas sont :

C7 =

D7 =

E7 =

A7 =

B7 =

4) Écriture d'un algorithme qui à partir de a=1 et b=2 exécute **8 fois le procédé** et affiche les bornes de l'encadrement obtenu :

a) D'après le tableau obtenu avec le tableur, quel résultat affichera votre algorithme ?

b) Décrire précisément la méthode.

Variante (3) : Sans passer par le I) on envisage directement la résolution de l'équation $x^2 - x - 1 = 0$ d'abord par lecture graphique, puis on retrouve lesquestions 2) et 3) du II) de l'activité proposée.

Ici a été introduit le texte à trous pour l'algorithme, malgré cela concevoir un algorithme été difficile.

Annexe 2 : commentaires – observations faites :

Le texte proposé a été élaboré à partir des variantes décrites en annexe (1), en tenant compte des difficultés rencontrées, mais n'a pas pu être testé tel quel ,par contre, chacune des variantes a été testée dans nos classes !!!!

Pré-requis algorithmique :

- Initiation, notions de variables, entrée sortie
- Notion de « si alors sinon »
- Notion de boucles « tant que.... »
- Calculatrice et programmation- premiers pas

Pré-requis mathématiques:

- Notion de fonction, de variations de fonctions, lectures graphiques de solutions d'équations
- Fonctions affines et fonction carrée.

Objectif : Mettre en place la notion de dichotomie.

Trouver une valeur approchée de solution d'équation par plusieurs méthodes : graphique, programmation,

Durée : dépend de la variante adoptée de l'initiative laissée aux élèves :

Texte proposé estimation : 2 séances d'une heure.

Variante 1 :(sur 3 séances) 5 minutes en fin de séance 1, séance 2 : 50 minutes en module , séance 3 : 20 minutes

Variante 2 : 3 séances d'une heure : tableur, recherche par les élèves d'un algorithme programmation sur algobox :

Variante 3 : (sur 2 séances) 1 séance d'heure en module, plus 30minutes sur une autre.

Déroulement : Variante 1 dans cette version il n'y avait pas le texte à trous

→ La veille de la séance de modules, classe entière, on joue les 5-10 dernières minutes du cours au jeu du nombre inconnu, plusieurs essais sont faits, je demande « qui a adopté une stratégie », qui « n'a pas adopté de stratégie ». La majorité a fait de la dichotomie 1/3 a répondu strictement au hasard, en constatant des difficultés à mémoriser les nombres qu'ils avaient déjà proposés.

→ Le lendemain en module, on reprend ce qui a été dit la veille en fin d'heure, on distribue la fiche et on refait le jeu comme il est indiqué sur la fiche

Un élève va au tableau et au fur et à mesure des réponses proposées, Il remplit le tableau suivant :

Etape n°	Nombre proposé	Intervalle dans lequel se situe le nombre à trouver
0		[0 ; 256]

A l'occasion des questions posées dans cette première partie, on peut définir « amplitude » d'un intervalle, centre d'un intervalle (le mot milieu a souvent été gardé en 2nde)

- Ils font seuls Le II)1°) : partie lecture graphique, sans trop de problèmes.
- Pour la partie II)2°)1) avec calculatrice, il a fallu ré expliquer à certains comment faire pour avoir le tracé d'une fonction, sinon pas de problème.
- Les difficultés qui se sont présentées au II)2°)2) pour remplir le tableau étaient surtout de type calculatoire : calcul du milieu, puis image de ce milieu, ils n'ont pas utilisé de façon systématique le programme fait antérieurement sur « calcul d'images par une fonction »

  Des graphiques sont apparus indispensables pour visualiser le changement des bornes de l'intervalle dans les différents cas, en insistant que l'on utilisait la croissance de la fonction sur [1 ; 2].

- A la fin de la séance de module, tous avaient traité le II) 2°) 2).

Certains étaient plus avancés et avaient complété l'algorithme du 3). Je demande à tous de faire le 3) a) pour la séance suivante, d'essayer de faire le b).

→ Séance suivante :

- On prend 20 minutes pour corriger et écrire le programme sur la TI 82, cela va assez vite car ils en ont déjà faits.

- On cherche la valeur exacte de la solution positive de l'équation à l'aide d'un changement d'écriture de f(x) .

Bilan et prolongements possibles:

Activité assez riche mais difficile, également testée dans sa variante 1 en 1S avec la résolution de cos(x) = x .

Il aurait été intéressant de demander de modifier l'algorithme pour une fonction décroissante sur l'intervalle. cela ne nous paraît pas indispensable pour ceux qui n'iront pas en 1S, et ces derniers auront sans doute l'occasion de le faire dans cette classe.

Avec la variante 1 :
La difficulté qui consiste à écrire un algorithme est gommée, on se contente de comprendre un algorithme donné, de le compléter.
L'intérêt est de pouvoir en une seule séance aller jusqu'à la programmation de l'algorithme sur calculatrice ou autre outil de programmation.
Avec la variante 2 : Utilisation du tableur et recherche par les élèves d'un algorithme :
Il faut avoir déjà pratiqué le tableur, les fonctions SI(test logique;valeur-si-vrai;valeur-si-faux);sinon on multiplie les difficultés pour les élèves.
Les élèves ont eu des difficultés à produire un algorithme, c'est pourquoi dans la fiche proposée, on a ajouté la question II)2°)3)a)
La variante 3 introduction d' un algorithme à trous, devant les difficultés des élèves à comprendre les changements de bornes pour les intervalles successifs, des graphiques ont été introduits dans la version finale que nous proposons.

Annexe 3 : productions d'élèves lors de la variante 2 : Là où on mesure les difficultés !!!!

Groupe 1	Groupe 2	Groupe 3
a=1 b=2 Je recommence 8 fois Je calcule la moyenne de a et b Je calcule f(m) Si l'image est inférieure à 0, a bouge et b ne bouge pas Si l'image est supérieure à 0, a ne bouge pas et b bouge J'affiche a et b	Prendre un nombre a et b que l'on désigne a et b On prend la moyenne de 1 et 2 On cherche leur image On répète ce procédé jusqu'à ce que l'on trouve le plus petit intervalle On affiche le résultat (le juste prix)	a vaut 1 b vaut 2 a+b/2=m m^2-m-1=f(m) a prend la valeur de m si f(m) est inférieur ou égal à 0 sinon a ne bouge pas b ne bouge pas si f(m) est inférieur ou égal à 0 sinon il prend la valeur de m on retourne à la ligne 3, 8 fois on affiche a et b
Groupe 4	**Groupe 5**	**Groupe 6**
On sait que le nombre d'or est compris entre 1 et 2. On prend donc 1.5 (la moitié) On calcule f(x) Si f(x) est négatif l'intervalle rétrécit Quand f(x) est négatif a prend la valeur de l'antécédent si f(x) est positif c'est b qui prend la valeur de l'antécédent. On recommence la boucle 8 fois	Entrée Le nombre a Le nombre b Le nombre m Initialisation Affecter valeur à a Affecter valeur à b >a Traitement Affecter valeur à m=(a+b)/2 Si f(a) est négatif alors on coupe entre a et b, si cette valeur est négative, on coupe de plus en plus proche de b Si f(a) est positif alors on coupe entre a et b, si cette valeur est négative, on coupe de plus en plus proche de a Sortie On affiche a et b et on obtient un encadrement du Er	On part de 1 et 2 On prend le milieu 1.5 On calcule son image C'est négatif donc on repart de 1.5 et 2 On prend le milieu 1.75 Je calcule son image, 1.75^2-1.75-1=2.35 donc positif Le nombre d'or est entre 1.5 et 1.75 On continue 8 fois
Groupe 7	**Groupe 8**	
Prendre un nombre a Prendre un nombre b Calculer la moyenne de a et b Si f(m)<0 alors a=m=1.5 et b=2 Si f(m)>0 alors a=1.5 et b=m=1.75 Répeter cela 8 fois Afficher a Afficher b A et b étant l'intervalle de Er	Variables a est du type nombre b est du type nombre m est du type nombre i est du type nombre début algorithme lire a lire b pour i allant de 1 à 8 on réduit l'écart en donnant des nombres de plus en plus rapproché on calcule 8 fois la moyenne de la moyenne. 8 fois en calculant l'image de m	

En classe de seconde

Simulation :

- Les chasseurs et les canards *(Calculatrice)*
- Lancer d'une pièce *(donné sous forme d'évaluation - intervalle de fluctuation)*
- Approximation de Pi par la méthode de Monte Carlo (*Algobox*)
- Le paradoxe du Grand Duc de Toscane

Simulation Les chasseurs et les canards

D'après un article de Daniel Vagost dans le bulletin 486 de l'APMEP

N chasseurs d'élite (ils ne ratent jamais leur cible) tirent au hasard simultanément sur N canards. Combien de canards sont-ils survivants après ces tirs ?

1) Simulation de cette expérience avec N = 4.

S'installer par groupe de 4 (cela fera 8 groupes).
Dans chaque groupe il y a donc 4 chasseurs. Au signal, chacun va choisir au hasard un entier entre 1 et 4, ce numéro correspondra au numéro du canard tué. On compte alors le nombre de canards survivants.

Chaque groupe recommence 20 fois. On obtient donc une série de 20 nombres représentant **le nombre de canards survivants** à chaque fois ; noter vos résultats ci-dessous :
……
……

Chaque groupe calcule la moyenne des canards survivants : M=……..

On met les résultats des 8 groupes en commun :
……..
……
Que peut-on dire de ces résultats ?
……..
……

La moyenne des canards survivants pour cet échantillon de taille 160 (20 fois 8) est donc……..

2) Mise en œuvre avec une calculatrice.

On va simuler 500 fois l'expérience avec la calculatrice:

a) Voici l'algorithme :

```
Pour i allant de 1 à 500                          (i est le numéro de l'expérience)
Les 4 canards sont vivants                        (on met quatre 1 dans la liste L1 )
    Pour j allant de 1 à 4                        (j est le n° du chasseur)
    Le chasseur n° j choisit au hasard un numéro A de 1à 4
    Le canard n°A est tué                         (dans L1(A) on met 0)
    Fin Pour
On compte le nombre de survivants de l'expérience n° i   (on met ces résultats dans L2)
Fin Pour
On affiche le nombre moyens de canards survivants.(en faisant la moyenne des nombres de L2)
```

Dérouler l'algorithme « à la main » pour bien le comprendre.

i = 1 Dans L1 il y a :…… j =1 A =….. Dans L1 il y a :…… j =2 A =….. Dans L1 il y a :…… j =3 A =….. Dans L1 il y a :…… j =4 A =….. Dans L1 il y a :…… Le nombre de survivants est :…….. Dans L2 il y a :……	**i = 2** Dans L1 il y a :…… j =1 A =….. Dans L1 il y a :…… j =2 A =….. Dans L1 il y a :…… j =3 A =….. Dans L1 il y a :…… j =4 A =….. Dans L1 il y a :…… Le nombre de survivants est :…….. Dans L2 il y a :……

Et on continue jusqu'à i = 500, puis on affiche le nombre moyens de canards survivants

b) Programmer cet algorithme avec votre calculatrice :

Programme à taper : Valider à la fin de chaque ligne	Où trouver les instructions :
PROGRAM : CANARD **: EFFLISTE L1, L2** **: FOR (I, 1,500)** **: SUITE (1, X, 1, 4) →L1** **: FOR (J, 1,4)** **: EntAlea (1, 4) →A** **: 0→L1(A)** **: End** **: Somme(L1) →L2(I)** **: End** **: Moyenne(L2)**	PRGM NOUV STAT 4 PRGM 4 Listes OPS 5 Pour la flèche : STO→ MATH PRB 5 PRGM 7 Listes MATH 5 LIST MATH 3
Exécuter le programme :	
prgmCANARD 1.276	On quitte le mode édition : Quitter PRGM EXEC (Il faut un certain temps à la calculatrice pour afficher le résultat.)

c) Remarque : Le calcul théorique (trop difficile pour nous pour l'instant) donne pour N = 4, $p = \frac{3^4}{4^3}$.

3) Transformer le programme pour simuler l'expérience avec N = 10.

Lancer d'une pièce- Exercice donné en évaluation en 2nde

On s'intéresse à l'expérience suivante : on lance trois fois une pièce de monnaie.
On simule cette expérience plusieurs fois et on note la fréquence de l'évènement « avoir au moins 2 piles ». On admettra que la fréquence théorique de cet évènement est $p = 0.5$.

I) Voici un algorithme qui simule plusieurs expériences :

Début de l'algorithme

- On initialise K à 0
- **Pour *i* allant de 1 à 100**
  a vaut 0 ou 1
  b vaut 0 ou 1
  c vaut 0 ou 1
  s est la somme de a, b et c
  Si s=2 **alors** on rajoute 1 à K **FinSi**
  Si s=3 **alors** on rajoute 1 à K **FinSi**
  Fin Pour
- On calcule f
- On affiche f

Fin de l'algorithme.

Questions :

1) Combien d'expériences sont simulées ?
……

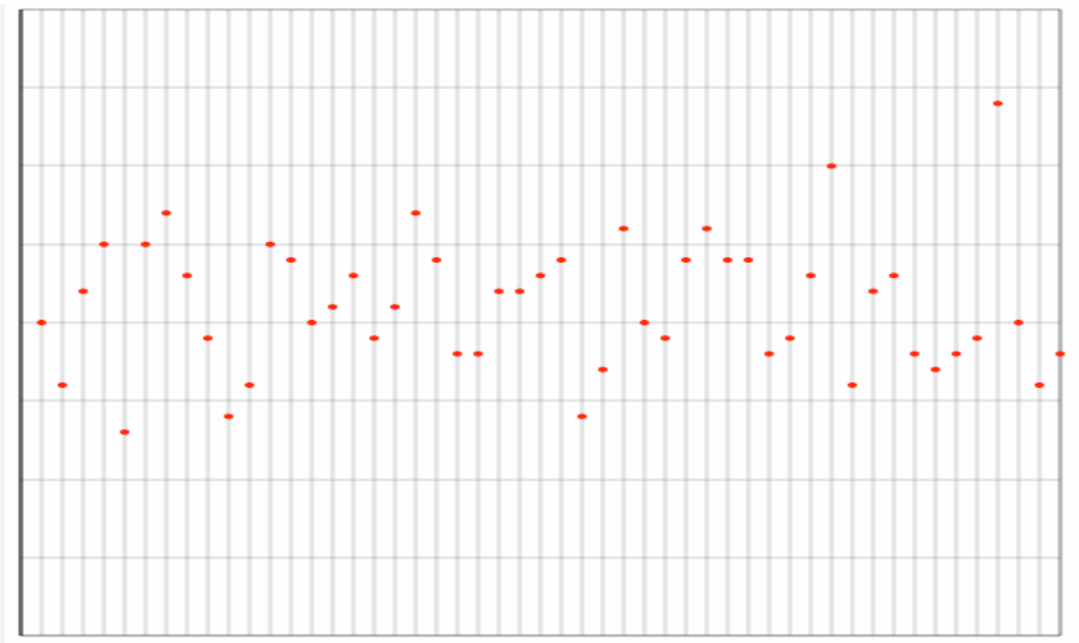
2) Que représente chaque variable?

……
……
……
……
……
……
……

3) Comment calculer f ?
……
……

4) Dans quel intervalle y a-t-il 95 % de chance de trouver la valeur f affichée ? Détailler les calculs.
……
……
……
……
……

II) On programme cet algorithme et on le teste 50 fois, voici les fréquences obtenues :



Questions :

1) Pourquoi les fréquences varient-elles ?

……
……

2) Donner la médiane de cette série :

……
……
……

3) Vrai ou faux ? Justifier.

a) Au moins 50% des fréquences sont supérieures à 0.51 ?

……
……
……
……

b) 95 % des fréquences appartiennent à l'intervalle [0.35 ; 0.65]»

……
……
……

c) Au moins 95 % des fréquences appartiennent à l'intervalle [0.35 ; 0.65]»

……
……
……
……

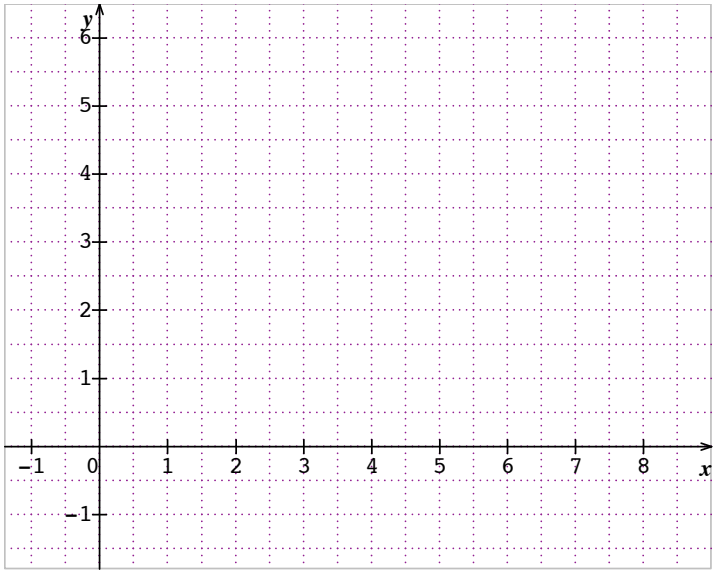
Plusieurs activités pour terminer par l'approximation de π par la méthode de Monte Carlo
avec ALGOBOX

Activité 1

Objectif : Ecriture un algorithme qui calcule la distance OM connaissant les coordonnées (x ; y) d'un point M du plan dans un repère orthonormé d'origine O

I) Le problème :

Soit le point de coordonnées (3 ; 4) dans un repère orthonormé d'origine O. Faire une figure et calculer la distance OM.



II) Ecriture de l'algorithme

Ecrire un algorithme qui calcule la distance OM connaissant les coordonnées (x ; y) d'un point M du plan dans un repère orthonormé d'origine O :

Quelles sont les variables à déclarer ?.....

Quelles sont les variables à lire ?.....

Quelle est la variable à calculer puis à afficher ?.....

Remarques : Pour calculer x^2 avec le logiciel Algobox, on écrit : pow(x,2)
Pour calculer une racine carrée, on écrit : sqrt(..)

Tester votre algorithme avec les valeurs $x = 3$ et $y = 4$.

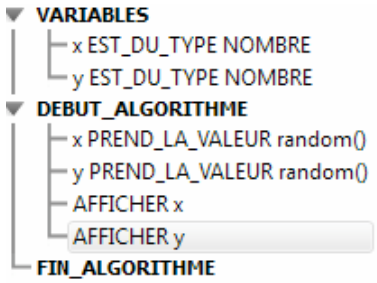
Activité 2

Objectif : OIAJ est un carré de côté 1. M est un point choisi au hasard de ce carré de coordonnées (x ; y) dans le repère ($(O,\overrightarrow{OI},\overrightarrow{OJ})$). Ecrire un algorithme qui teste si le point M est à l'intérieur ou non du disque de centre O et de rayon 1.

I) Choisir un nombre au hasard :

Avec la calculatrice TI, taper MATH, PRB puis NbrAléat
Qu'obtient_on ?
.....
.....

Ecrire avec Algobox ce programme, le tester. Que fait-il ?.....



II) Ecriture de l'algorithme :

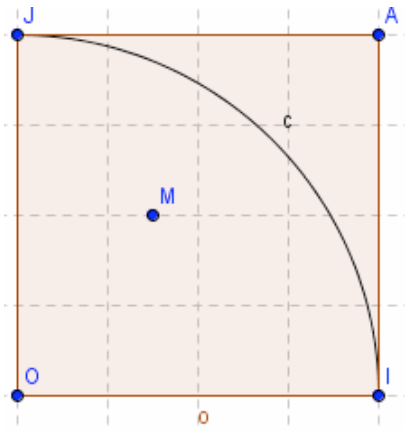
OIAJ est un carré de côté 1. M est un point choisi au hasard de ce carré de coordonnées (x ; y) dans le repère ($(O,\overrightarrow{OI},\overrightarrow{OJ})$).

Notons d la distance OM.

Si M est un point du quart de cercle tracé, combien vaut d ?.....

Comment tester si le point M est à l'intérieur ou non du disque de centre O et de rayon 1 ?

……

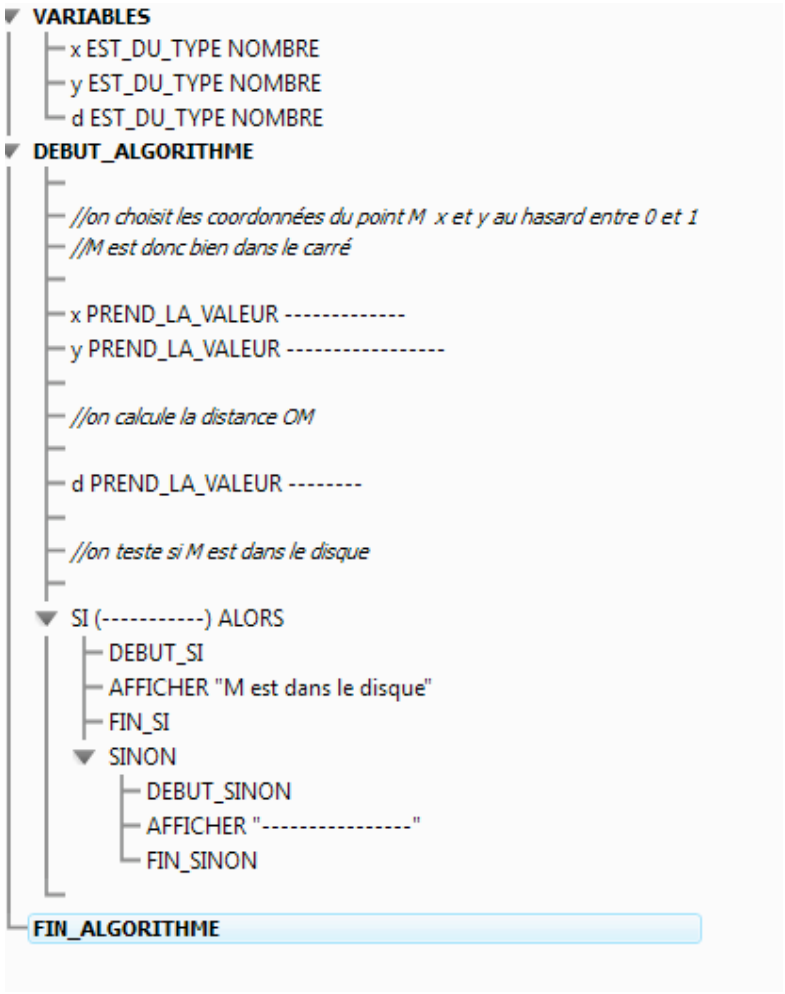


Exprimer d en fonction de x et y :

……
……

Compléter l'algorithme suivant qui teste si le point M est à l'intérieur ou non du disque de centre O et de rayon 1.

En italique sont notés les commentaires



Activité 3

Objectif : Approximation de π par la méthode de Monte Carlo

1) De quoi s'agit-il ?

Soit un point Mde coordonnées (x,y), où $0<x<1$et $0<y<1$. On tire aléatoirement les valeurs de x et y.

Si $x^2+y^2<1$ alors le point M appartient au disque de centre $(0,0)$de rayon 1.

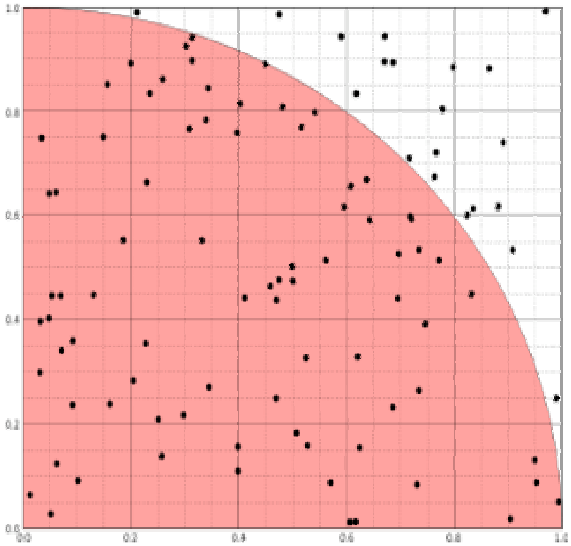
On recommence avec plusieurs points (1000 par exemple). On compte le nombre de points à l'intérieur du disque, notons C ce nombre.

En faisant le rapport du nombre de points dans le disque par rapport au nombre de tirages, si le nombre de tirages est grand, on obtient une approximation du quotient de la surface du quart de disque avec la surface du carré.

Quelle est la surface du carré ?.....

Quelle est la surface du quart de disque ?.....

Donner alors en fonction de C une approximation de π :……..



Faire des essais : http://www-sop.inria.fr/mefisto/java/tutorial1/node15.html

Le véritable développement des méthodes de Monte-Carlo s'est produit lors de la 2nde guerre mondiale, lors des recherches sur la fabrication de la bombe atomique, sous l'impulsion de von Neumann et Ulam.

La dénomination *Méthode de Monte-Carlo* provient de ce que les meilleures suites de nombres aléatoires sont données par la roulette des casinos...

2) **Ecriture de l'algorithme**

- On prend un point M au hasard dans le carré
- On teste s'il est ou non dans le disque
- On recommence 1000 fois en comptant combien de points sont dans le disque
- On en déduit une approximation de π

Le paradoxe du Grand Duc de Toscane

A partir d'une activité proposée par l' APMEP de la réunion

1) Contexte historique

Galilée (1554-1642) est surtout connu pour ses travaux en astronomie, faisant suite à son invention de la lunette astronomique. Cependant, il rédigea vers 1620 un petit mémoire sur les jeux de dés pour répondre à une demande du Duc de Toscane (Galilée est alors Premier Mathématicien de l'Université de Pise et Premier Philosophe du Grand Duc à Florence). Galilée est ainsi l'un des premiers avec Cardan à avoir écrit sur le "calcul des hasards", mais leurs écrits n'ont été publiés qu'après la célèbre correspondance entre Pascal et Fermat qui marque "officiellement" le début de la théorie des probabilités. Le mémoire de Galilée qui nous intéresse n'a été édité qu'en 1718.

2) Présentation du paradoxe

A la cour de Florence, de nombreux jeux de société étaient alors pratiqués. Parmi ceux-ci, l'un faisait intervenir la somme des numéros sortis lors du lancer de trois dés. Le Duc de Toscane, qui avait sans doute observé un grand nombre de parties de ce jeu, avait constaté que la somme 10 était obtenue légèrement plus souvent que la somme 9.
Le paradoxe, que le Duc avait exposé à Galilée, réside dans le fait qu'il y a autant de façons d'écrire 10 que 9 comme sommes de trois entiers compris entre 1 et 6 :
Vérifier cette dernière affirmation :

……
……
……
……

3) Simulation de 1000 expériences avec le tableur

Le problème du duc de toscane

lancer n°	1er dé	2ième dé	3ième dé	somme
1	3	6	3	12
2	6	5	5	16
3	2	6	4	12
4	5	1	3	9
5	6	3	2	11
6	5	5	6	16
7	2	2	1	5
8	2	4	2	8
9	3	2	2	7
10	4	1	5	10
et	3	1	5	9
12	5	4	6	15
/D7)	5	2	1	8
14	5	1	2	8
15	1	3	6	10
16	5	6	2	13
17	5	5	2	12
18	4	2	6	12

Nombre de 9	119
Nombre de 10	130

Formules :

en B7 : =ALEA.ENTRE;BORNES(1;6)
à recopier colonnes B, C et D.

en E7 : =SOMME(B7:D7)

en G8 : =NB.SI(E7:E1006 ; « =9 »)

en G10 : : =NB.SI(E7:E1006 ; « =10 »)

Pour effectuer plusieurs simulations, on appuie sur Maj Ctrl F9.
Pourquoi obtient-on des fréquences différentes à chaque fois ?.....

On peut augmenter la taille de l'échantillon en recopiant les formules plus bas. Qu'obtient-on ?.....
Retrouve-t-on l'observation du duc de toscane ?.....

4) Simulation de l'expérience avec l'écriture d'un algorithme.

Ecriture de l'algorithme :

On décide d'avoir deux compteurs : X pour les sommes égales à 9 et Y pour les sommes égales à 10.
On initialise X et Y, à 0
On répète 1000 fois :
    Choix d'une valeur A pour le 1er dé
    Choix d'une valeur B pour le 2ième dé
    Choix d'une valeur C pour le 3ième dé
    On fait la somme S des trois nombres choisis
    Si S=9 alors on ajoute 1 à X
    Si S=10 alors on ajoute 1 à Y
Fin de boucle
On affiche la fréquence de 9, c'est-à-dire X/1000
On affiche la fréquence de 10, c'est-à-dire Y/1000

Remarque : on peut aussi faire en sorte que l'algorithme demande la valeur de N.

Programmation avec la TI :

```
PROGRAM:TOSCANE
:0→X
:0→Y
:For(I,1,1000,1)
:randInt(1,6)→A
:randInt(1,6)→B
:randInt(1,6)→C
:A+B+C→S
:If S=9
:Then
:X+1→X
:End
:If S=10
:Then
:Y+1→Y
:End
:End
:Disp X/1000
:Disp Y/1000
:
```

```
prgmTOSCANE
                .134
                .147
                Done
```

Programmation avec Algobox :

```
DEBUT_ALGORITHME
  x PREND_LA_VALEUR 0
  y PREND_LA_VALEUR 0
  POUR i ALLANT_DE 1 A 10000
    DEBUT_POUR
    a PREND_LA_VALEUR floor(random()*6+1)
    b PREND_LA_VALEUR floor(random()*6+1)
    c PREND_LA_VALEUR floor(random()*6+1)
    s PREND_LA_VALEUR a+b+c
    SI (s==9) ALORS
      DEBUT_SI
      x PREND_LA_VALEUR x+1
      FIN_SI
    SI (s==10) ALORS
      DEBUT_SI
      y PREND_LA_VALEUR y+1
      FIN_SI
    FIN_POUR
  fx PREND_LA_VALEUR x/10000
  fy PREND_LA_VALEUR y/10000
  AFFICHER fx
  AFFICHER fy
FIN_ALGORITHME
```

Résultats

```
***Algorithme lancé***
0.1206
0.1249
***Algorithme terminé***
```

5) Elucidation

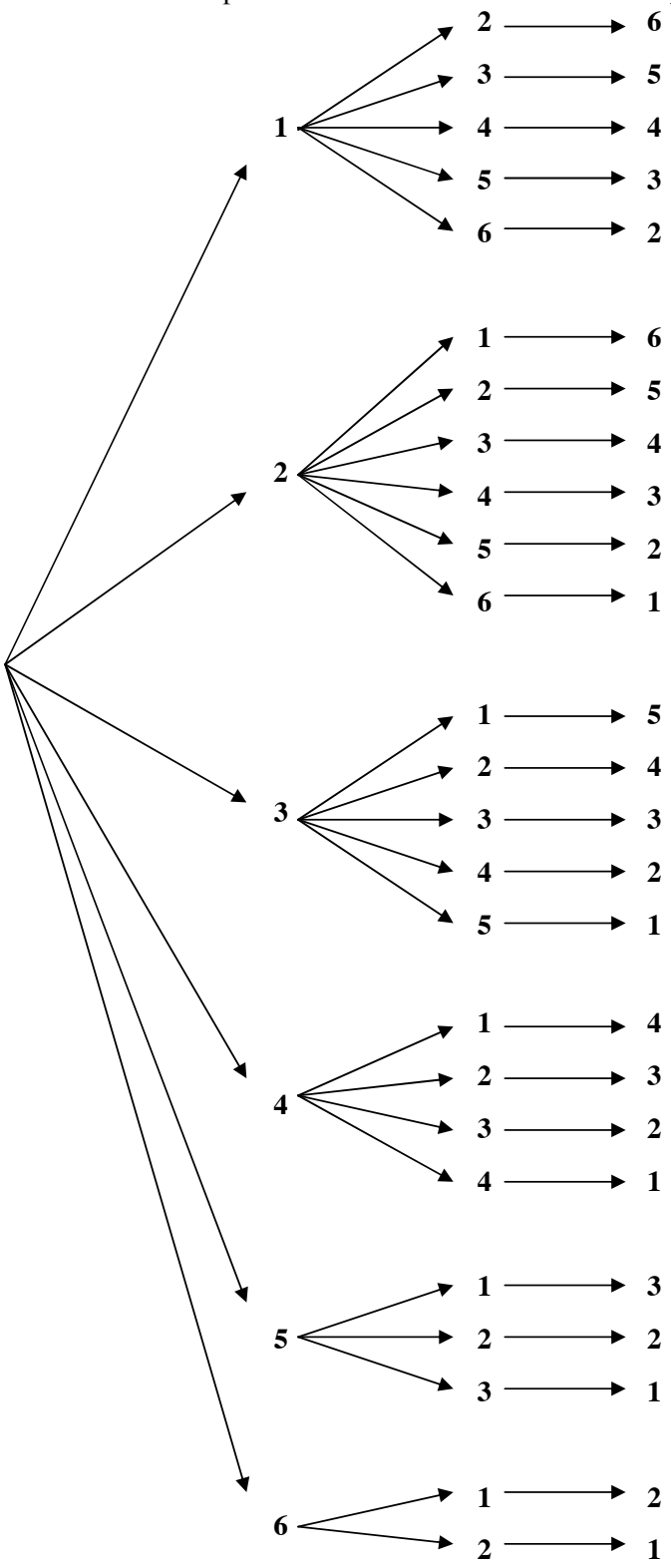
Cas possibles :

Pour faire une étude théorique, on va dénombrer toutes les possibilités des lancers de trois dés :

On a …….choix pour le numéro obtenu pour le 1^{ier} dé, une fois ce numéro choisi on a ……choix pour le 2^{ième} dé et une fois ce deuxième numéro choisi on a ……choix pour le 3^{ième} dé. Au final on a donc……choix pour ces triplets de nombres.

Cas favorables :

L'arbre ci-dessous permet de dénombrer les différentes façons d'obtenir une somme égale à 9 :



Quelle est donc la probabilité d'obtenir une somme égale à 9 ?.....

A l'aide de l'arbre précédent, à compléter, trouver la probabilité d'obtenir une somme égale à 10.
……..

Dans ces calculs de probabilité, on a utilisé une formule du cours qui est valable dans quelles conditions ?.....

Expliquer le paradoxe du grand duc de toscane :

……
……
……

5) A retenir

Le paradoxe vient du fait que les possibilités dénombrées par le Grand Duc ne sont pas équiprobables : une somme comme 3 + 3 + 3 a trois fois moins de chance d'être obtenue qu'une somme comme 5 + 2 + 2 , et six fois mois qu'une somme comme 4 + 3 + 2 .

Les probabilités d'obtenir une somme égale à 9 ou à 10, sont respectivement 25/216 et 27/216, soit 0,116 (environ) et 0,125.

Attention au choix d'un univers sur lequel l'hypothèse d'équiprobabilité des issues puisse être admise ou non.

Cette question est loin d'être évidente et sa résolution a d'ailleurs rencontré historiquement de sérieux atermoiements comme l'atteste l'article "*croix ou pile*", pourtant beaucoup plus tardif, de d'Alembert dans l'Encyclopédie (publiée entre 1751 et 1772).

En Première

- Suite de Syracuse *(activités d'initiation aux suites numériques- Algobox)*
- Fiches d'exercices sur les suites *(Lectures d'algorithmes)*
- Valeur approchée du nombre d'or par la méthode des tangentes. *(Calculatrice)*
- Le paradoxe du Grand Duc de Toscane *(Introduction aux probabilités) Voir fiche de la partie « Simulation » en Seconde.*
- Espérance d'une variable aléatoire (*Calculatrice*)
- Accroissement ponctuel (*Nombre dérivé / calculatrice*)
- Les bonbons (*Evaluation / Algobox – calculatrice*)

TP -1ES-Suite numériques-Activité avec Algobox- **Suite de Syracuse**

1) De quoi s'agit-il ?

Algorithme de Collatz :

- Choisir un nombre entier *a*.
- Si le nombre est pair, on le divise par 2 et on obtient un nouveau nombre.
- Si le nombre est impair, on le multiplie par 3, on ajoute 1 au résultat et on obtient un nouveau nombre.

- On recommence la procédure avec le nouveau nombre obtenu.

Faire quels essais : avec $a = 5$ puis $a = 12$ en recommençant le procédé 10 fois.

.....
.....
.....
.....

.....
.....
.....
.....

On obtient une suite de nombres qui est appelée **le vol** du nombre de départ, les nombres de la suite sont appelés **les étapes** du vol, le plus grand nombre obtenu dans la suite est appelé **l'altitude maximale** du vol et le nombre d'étapes avant d'obtenir 1 est appelé **la durée** du vol.

Conjecture de Syracuse :
(Une conjecture est une supposition, une affirmation qui n'est pas démontrée)

On finit toujours par obtenir 1 dans la suite obtenue avec l'algorithme de Collatz avec n'importe quel nombre au départ.

On peut faire des essais : http://trucsmaths.free.fr/js_syracuse.htm

2) Notation indicielle

Pour ***a* = 7** on obtient la suite de nombres : 7 – 22 – 11 – 34 – 17 -…..etc.
on note $U_0 = 7$ $U_1 = 22$, $U_2 = 11$…. U_n est le terme de rang *n* ou d'indice *n*.
L'ensemble de tous les nombres est une suite notée (U_n).
Combien vaut U_5 ? U_6 ?.....
Le terme suivant U_n est U_{n+1}. Comment obtient-on U_{n+1} à partir de U_n ?
.....
.....
.....

Comment pourrait-on calculer U_{20} ? .....

Cette suite est donc définie par son premier terme U_0 et une relation entre U_{n+1} et U_n appelé relation de récurrence.

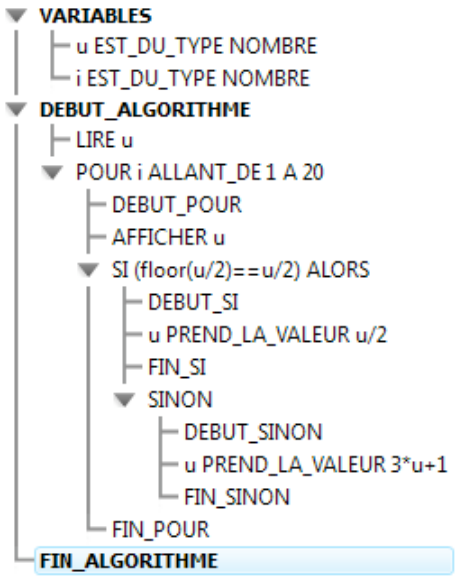
3) Algorithme de calcul

a) On peut écrire un algorithme qui affiche les 20 premiers termes de cette suite à partir de la donnée du premier terme :

On demande U la valeur du premier terme de la suite
On fait 20 fois :
- *Afficher U*
- *Si U est pair alors on remplace U par U/2 sinon par 3*U+1*

Pour tester la parité d'un nombre *a* :
a est pair si c'est un multiple de 2, c'est-à-dire si le quotient de *a* par 2 est encore un nombre entier. Cela est équivalent de dire que la partie entière de *a*/2 est égal à *a*/2.

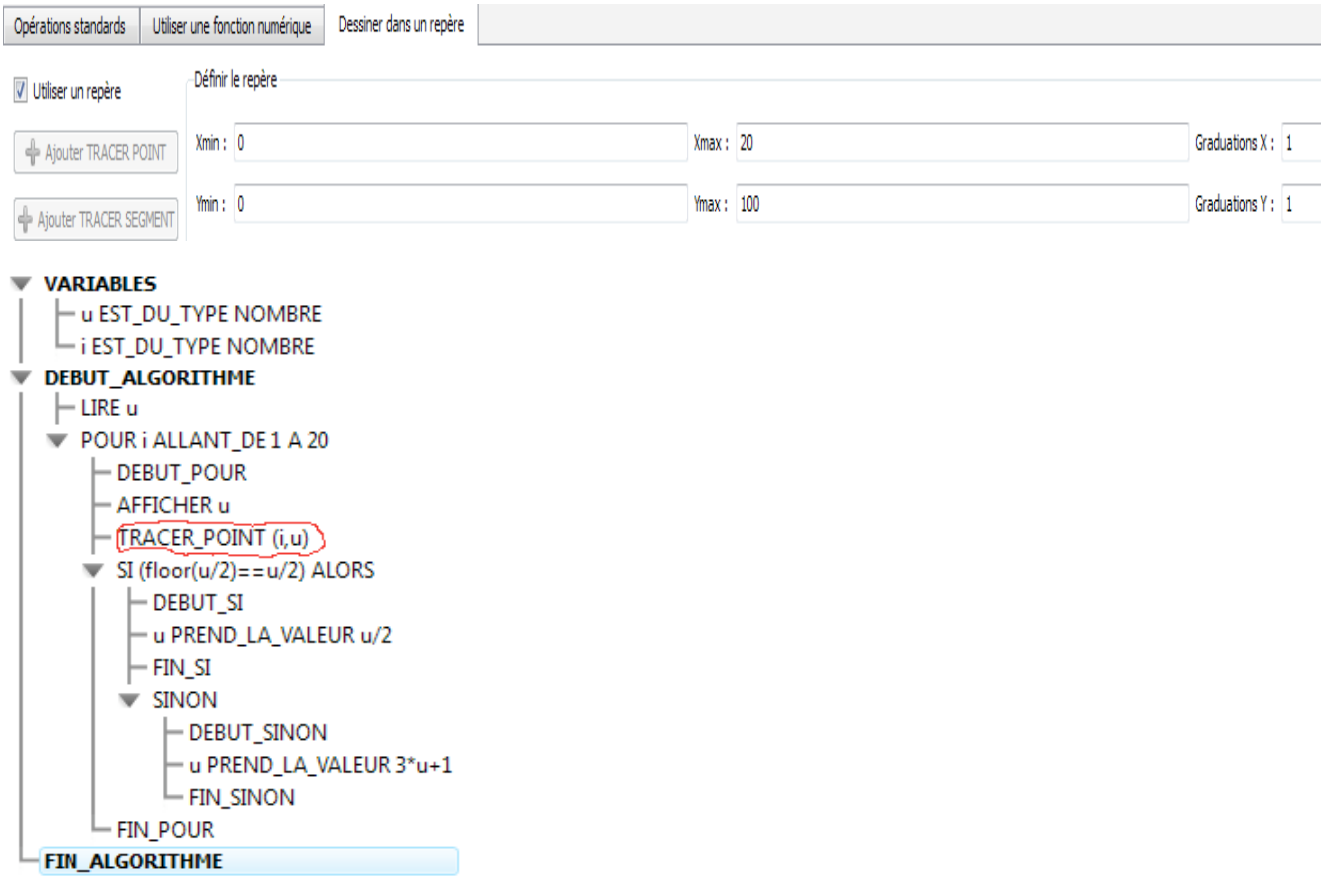
On peut maintenant programmer cet algorithme :
Avec Algobox cela donne :



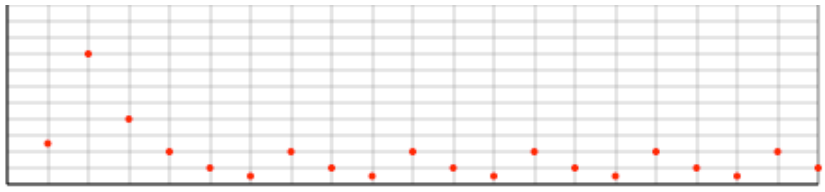
Utiliser votre programme pour compléter le tableau suivant :

U_0	Les 20 premiers termes de la suite (U_n)
1	1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4
2	2 ; 1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1
3	
4	4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;1
5	5 ;16 ;8 ;4 ;2 ;1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2
6	6 ; 3 ;10 ;5 ;16 ;8 ;4 ;2 ;1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2
7	
8	
9	9 ;28 ;14 ;7 ; 22 ;11 ;34 ;17 ;52 ;26 ;13 ;40 ;20 ;10 ;5 ; 16 ;8 ;4 ;2 ;1
10	10 ; 5 ;16 ;8 ;4 ;2 ;1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4
16	16 ;8 ;4 ;2 ;1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1
21	

On peut aussi avec Algobox, représenter graphiquement cette suite en définissant d'abord un repère :



On obtient, avec 5 comme premier terme, le graphe suivant : (Rajouter le repère).



On retiendra que la représentation graphique d'une suite (U_n) est l'ensemble des points de coordonnées (n, U_n).
n est un entier naturel. On ne relie pas les points.

b) Ecrire l'algorithme qui affiche cette suite de nombres, avec pour condition d'arrêt l'obtention du chiffre 1.

c) Incorporer à ce programme un compteur qui détermine la durée du vol.

d) Faire ensuite afficher l'altitude du vol.

e) Compléter le tableau ci-dessous :

U_0	p	altitude
1		
2		
3		
4		
5		
6	9	16
7		
8		
9		
10		

4) Des suites particulières

Compléter le tableau suivant :

U_0	Les termes de la suite (U_n) jusqu'à l'obtention de 1
2	2 ; 1
4	4 ; 2 ;1
8	
16	

Quelle remarque peut-on faire sur les « U_0 choisis » ? .....

Dans ces cas particuliers, quelle est la relation de récurrence entre U_{n+1} et U_n ?.....

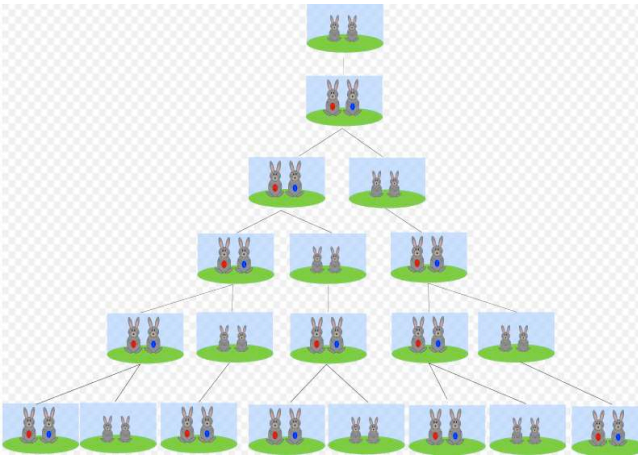
A l'aide du logiciel, représenter graphiquement ces suites.

Supposons que U_0 = 1024. Comment calculer U_6 à partir de U_0 ?

.....

.....

I) La suite de Fibonacci.



```
VARIABLES
    a EST_DU_TYPE NOMBRE
    b EST_DU_TYPE NOMBRE
    c EST_DU_TYPE NOMBRE
    i EST_DU_TYPE NOMBRE
    n EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE n
    a PREND_LA_VALEUR 1
    b PREND_LA_VALEUR 1
    POUR i ALLANT_DE 1 A n-1
        DEBUT_POUR
        c PREND_LA_VALEUR a+b
        a PREND_LA_VALEUR b
        b PREND_LA_VALEUR c
        FIN_POUR
    AFFICHER c
FIN_ALGORITHME
```

1) Tester cet algorithme pour $n = 5$ en remplissant le tableau suivant :

	a =	b =	c =
Valeurs initiales			
i=1			
i=2			
i=3			
i=4			

En déduire la valeur affichée par l'algorithme pour n =5 ?

2) Quelle valeur est affichée pour n =10 ?

3) Soit la suite définie par $U_0 = 1$, $U_1 = 1$ et pour n >1, U_n le résultat affiché par l'algorithme ci-dessus. Déterminer une relation de récurrence définissant U_n à partir des termes le précédant.

II)

Associer à chaque exercice l'algorithme qui permet de répondre à la question posée et écrire un exercice correspondant à l'algorithme restant :

1 : Un capital de 2000 € est placé à 3 % à intérêt composés. Quel sera le capital acquis au bout de 10 ans ?

2 : Un capital de 2000 € est placé à 3 % à intérêt composés. Au bout de combien d'année le capital acquis dépassera 3000 € ?

3 : Calculer la somme des 11 premiers termes de la suite géométrique de raison 1.03 et de premier terme 2000.

A	B

u = 2000
s = u
Pour i allant de 1 à 10
  u = u × 1.03
  s = s+u
fin pour
afficher s
	u = 2000
	Pour i allant de 1 à 10
	  u = u × 1.03
	fin pour
	afficher u

C	**D**

u = 2000
s = 0
Tant que u ≤ 3000
  u = u × 1.03
  s = s+1
fin tant que
afficher s
	u = 2000
	s = u
	Tant que u ≤ 3000
	  u = u × 1.03
	  s = s+u
	fin tant que
	afficher s

1L Valeur approchée du nombre d'or par la méthode des tangentes.

f étant la fonction définie $f(x) = x^2 - x - 1$, la solution positive de l'équation $f(x) = 0$ est appelée nombre d'or et notée φ.

1) Déterminer les variations de f.
2) Tracer la parabole (P) représentant f puis donner un encadrement de φ par deux entiers.
3) Calculer l'équation de la tangente (T_a) à (P) en son point d'abscisse $a = 2$.
4) Calculer l'abscisse b du point d'intersection de (T_a) avec l'axe des abscisses.

L'idée est de prendre b comme valeur approchée de φ puis de recommencer le procédé, c'est-à-dire de tracer la tangente (T_b) à (P) en b puis de prendre l'abscisse du point d'intersection de (T_b) avec l'axe des abscisses comme meilleure approximation du nombre φ.

5) Recommencer encore le procédé à partir de b et donner une nouvelle valeur approchée de φ.
6) Ecrire un algorithme qui donne une valeur approchée de φ après 10 étapes de ce procédé.
7) Programmer cet algorithme avec votre calculatrice.
8) Déterminer la valeur exacte de φ.

Compte-rendu rapide

Exercice fait en 1h avec 1L
Prérequis :
Algorithmique : Les élèves ont déjà beaucoup manipulé toutes les notions au programme.
Mathématiques : Le chapitre dérivation est terminé.

Intérêt : L'exercice se traite bien en une heure et le programme à écrire avec la TI est court.

Voici l'algorithmique 1 proposé par les élèves :

```
a vaut 2
Pour i allant de 1 à 10
   On détermine l'équation de la tangente en a
   On cherche b
   On remplace a par b
Fin pour
Afficher a
```

Le problème pour eux est alors de transcrire cet algorithme en langage machine.

Comment calculer b ?

L'équation de T_a est $y = f'(a)(x - a) + f(a)$

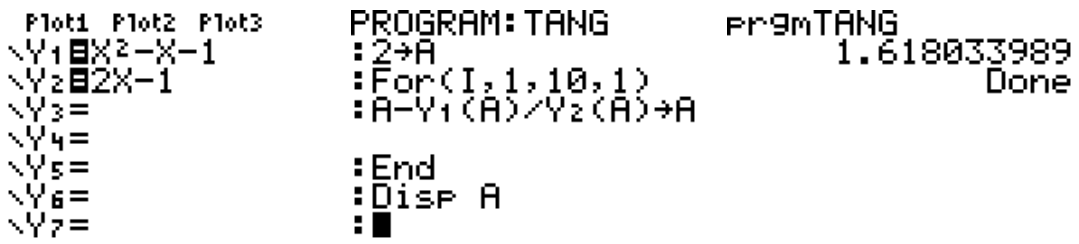
Donc $0 = f'(a)(b - a) + f(a)$ d'où $b = a - \dfrac{f'(a)}{f(a)}$

Ils se rendent compte alors que l'algorithme peut être simplifié :

Algorithme 2 :

```
On initialise a à 2
Pour i allant de 1 à 10
   On remplace a par b
Fin pour
Afficher a
```

Programme avec la TI :



Une variable aléatoire et son espérance

Un joueur lance deux fois de suite une pièce de monnaie équilibrée.

Un résultat de cette expérience aléatoire est un couple : par exemple on peut obtenir Pile puis Face, ce qui se note (Pile ; Face).

1° Quel est l'univers E de cette expérience ? E = { ……}

2° Etablir la loi de probabilité associée à cette expérience.

……

……

Le joueur gagne 1 € par pile obtenue et il perd 2 € par face obtenue.

3° a) Quel est le gain algébrique du joueur quand il obtient (Pile ; Face) ? Un gain algébrique peut être positif ou négatif.

……

On désigne par X le gain algébrique (positif ou négatif) du joueur

b) Quelles sont toutes les valeurs possibles pour X ? ……

c) Compléter le tableau ci-dessous :

Gain x_i	x_1 = ……	x_2 =……	x_3 =……
P (X = x_i)			

Lorsqu'à chaque issue d'une expérience aléatoire, on associe un nombre réel, on dit qu'on définit une **variable aléatoire**. Ici, à chaque issue, on associe le gain X correspondant : X est une variable aléatoire.

4° Grâce à la calculatrice, on va simuler un grand nombre de parties pour avoir une estimation de ce que peut espérer gagner un joueur à ce jeu ...

a) La calculatrice peut choisir au hasard des nombres. Comment faire pour simuler le lancer d'une pièce en utilisant la calculatrice ? ……

b) Compléter l'algorithme si dessous pour qu'il simule deux lancers d'une pièce et pour qu'il affiche le gain.

Variables
A, B, X : nombres
Début
A prend aléatoirement ……
B prend aléatoirement ……

Afficher X
Fin

c) Que faudrait-il ajouter à cet algorithme pour simuler non plus une mais 800 parties de ce jeu ?

……

d) Compléter ci-dessous l'algorithme pour qu'il simule 800 parties et pour qu'il affiche le gain algébrique total puis le gain algébrique moyen par partie.

Variables
A, B, S, M,..... : nombres *// S est le gain algébrique total, M est le gain algébrique moyen*
Début
S prend la valeur 0

Afficher S

Afficher M
Fin

e) Programmer cet algorithme sur la calculatrice

Pour choisir aléatoirement 0 ou 1 :

- Avec la TI 82 stat.fr : **entAléa(0,1)** — math puis PRB puis entAléa(
- Avec la TI 82 : **int (rand × 2)** — MATH puis NUM choisir int puis MATH puis PRB choisir rand
- Avec la casio : **Int(Ran#× 2)** — OPTN puis NUM choisir Int puis EXIT choisir PROB puis choisir Ran #

f) Quel gain moyen obtenez-vous pour 800 parties jouées ? .....

5° Sur 800 parties jouées, combien de fois devrait-on , d'un point de vue probabiliste, avoir un gain de - 4 € ? .....

Même question avec -1 € : ..... puis avec 2 € : .....

En déduire l'espérance de gain sur 800 parties jouées : .....

Puis l'espérance de gain sur une partie : .....

Si on joue N parties de ce jeu, quel "gain" peut-on espérer ? .....

Le nom "**espérance mathématique**" est issu du langage des jeux.
L'espérance de gain s'interprète comme la moyenne des gains obtenus en répétant le jeu un grand nombre de fois.
Le jeu est **favorable** au joueur si son espérance de gain est strictement positive et **défavorable** au joueur si c'est un nombre strictement négatif. Lorsque l'espérance est nulle, le jeu est dit **équitable**.

Exercice

Une association sportive organise une loterie. Les 2000 billets vendus sont numérotés de 1 à 2000.
Parmi tous les billets, huit billets sont gagnants : un billet rapporte 1500 € , deux billets rapportent 150 € chacun , cinq billets rapportent 100 € chacun. Tous les autres billets sont perdants.
Le prix du billet est fixé à 2 €.
Quel est l'espérance de gain d'une personne achetant un billet ? Cette loterie est-elle équitable ?

Compte rendu de l'activité Variable aléatoire et espérance

Niveau : Cette activité a été faite en première S et pourrait aussi être réalisée en première ES

Durée : une séance de 55 minutes environ. Cette activité a été testée en demi-classe.

Objectif : Découvrir la notion de variable aléatoire et la notion d'espérance.

Prérequis : Le cours de seconde sur les probabilités, maîtrise des notions algorithmiques

Déroulement :

Les trois premières questions ne posent pas de problème.

En ce qui concerne la question 4°: si on a déjà travaillé la simulation de lancers de dés, les élèves pensent facilement à attribuer aléatoirement la valeur 0 ou 1 pour simuler le lancer d'une pièce (certains attribuent les valeurs 1 ou 2)

On adopte par exemple la convention suivante : 1 simule face et 0 simule pile.

Avant de passer à l'algorithme, on peut faire le point avec eux :

A est la première valeur aléatoire obtenue et B la deuxième.

Voilà les 4 différents cas de figure :

Si A = 1 et si B = 0 alors X = - 1

Si A = 0 et si B = 1 alors X = - 1

Si A = 1 et si B = 1 alors X = - 4

Si A = 0 et si B = 0 alors X = 2

Pour les aider, on peut leur demander comment faire pour ne plus avoir que trois cas.

Dans les deux groupes, des élèves ont pensé à faire la somme A + B.

Ce qui simplifie l'algorithme de la question 4 b) :

Si A + B = 1 alors X prend la valeur - 1

Si A + B = 2 alors X prend la valeur - 4

Si A + B = 0 alors X prend la valeur 2

Quant à la question d), la difficulté est de penser à "cumuler" les gains : S prend la valeur S – 1 ou S – 4 , ou S + 2

Le reste de l'activité ne me semble pas avoir posé problème.

TP Accroissement ponctuel

Première S3

L'objectif du TP est de tenter de répondre à la question suivante :
« *Quelle est la valeur exacte de l'accroissement de la fonction racine carrée en $x = 3$?* »

Notons f la fonction racine carrée, c'est-à-dire $f(x) = \sqrt{x}$ pour $x \in [0\,;+\infty[$.

Partie A – Estimation par la méthode du cours

1. Calculez une valeur approchée de $\mathrm{d}f/\mathrm{d}x$ en $x = 3$?

Pour calculer une valeur approchée de $\mathrm{d}f/\mathrm{d}x$ vous avez choisi une petite valeur pour h et vous avez effectué le calcul :

$$\frac{\sqrt{3+h}-\sqrt{3}}{h}$$

La valeur obtenue dépend donc du choix de h.

Le cours nous dit que si le calcul se rapproche d'une certaine valeur lorsque l'on choisit h de plus en plus petit alors cette valeur est la valeur exacte de l'accroissement ponctuel. Dans le cas contraire, l'accroissement ponctuel n'existe pas.

Nous allons effectuer ce calcul pour des valeurs de h de plus en plus petites : 0,1 ; 0,01 ; 0,001 ; etc.

2. Le texte ci-dessous décrit un programme qui permet d'automatiser les calculs avec votre calculatrice.

Vider la liste L_1
Vider la liste L_2
Pour N allant de 0 jusqu'à 13
  Mettre la valeur 10^{-N} dans H
  Ajouter la valeur de H à la liste L_1
  Calculer $\left(f(X+H)-f(X)\right)/H$
  Ajouter le résultat à la liste L_2
fin du pour

(a) Écrivez ci-dessous le programme à saisir dans votre calculatrice. Éxécutez le et mettez le au point en vérifiant en particulier qu'il redonne bien la valeur que vous aviez trouvée à la première question du TP.

(b) Présentez les valeurs calculées par votre programme dans un tableau en écrivant toutes les décimales fournies par la calculatrice.

3. Examinez les valeurs du tableau. Notez ce que vous remarquez. À votre avis, l'accroissement ponctuel existe-t-il en $x = 3$? Si oui, quelle est la meilleure valeur approchée que l'on ait obtenue.

Partie B – Estimation par une autre méthode

Le texte ci-dessous est extrait du manuel de votre calculatrice. Il décrit le rôle de la commande `nbreDérivé(`. Cette commande est accessible par les touches [math] 8.

- *Comme il est dit dans le cours, l'expression* nombre dérivé *est synonyme d'*accroissement ponctuel*;*
- *La notation $f'(x)$ (remarquez bien l'apostrophe) se lit « f prime de x ». Elle est équivalente à la notation $\mathrm{d}f/\mathrm{d}x$: elle désigne l'accroissement ponctuel de f en x.*

nbreDérivé(

nbreDérivé((nombre dérivé) donne une valeur approximative de la dérivée de l'*expression* par rapport à la *variable*, au point *valeur* ; la précision est liée à ε (si pas déterminé, la valeur par défaut est 1E-3).

nbreDérivé(*expression*,*variable*,*valeur*[,ε])

nbreDérivé(fait appel à la méthode de la dérivée symétrique qui donne une approximation du nombre dérivé par la pente d'une sécante.

$$f'(x) = \frac{f(X+\varepsilon)-f(X-\varepsilon)}{2\varepsilon}$$

A mesure que ε diminue, l'approximation devient plus précise.

```
nbreDérivé(A^3,A
,5,.01)
                75.0001
nbreDérivé(A^3,A
,5,.0001)
                     75
```

nbreDérivé(ne peut être utilisée qu'une seule fois dans une *expression*. En raison de la méthode appliquée pour calculer **nbreDérivé(** , la TI-82 Stats.fr peut donner une valeur dérivée fausse en un point où t n'est pas dérivable.

1. **(a)** Utilisez cette commande pour estimer $\mathrm{d}f/\mathrm{d}x$ en $x = 3$.
Écrivez la commande que vous avez saisie et le résultat que vous avez obtenu :

(b) La méthode d'estimation de l'accroissement ponctuel utilisée par la commande `nbreDérivé(` est-elle identique à celle vue dans votre cours ? Sinon, comparez les et expliquez les différences :

2. **(a)** Modifiez le programme que vous avez écrit précédemment de façon à ce qu'il estime $\mathrm{d}f/\mathrm{d}x$ en $x = 3$ selon la méthode de la commande `nbreDérivé(`.

(b) Présentez les valeurs calculées par le nouveau programme dans un tableau en écrivant toutes les décimales fournies par la calculatrice.

(c) La méthode de la commande `nbreDérivé(` est-elle plus efficace ?

Partie C – Ridiculisons la calculatrice

Comment ? En trouvant de façon *irréfutable* la valeur *exacte* de $\mathrm{d}f/\mathrm{d}x$ en $x = 3$.
Reprenons le problème à la base. Il s'agit de savoir si la quantité

$$\frac{\sqrt{3+h}-\sqrt{3}}{h}$$

se rapproche d'une valeur particulière lorsque l'on choisit h de plus en plus petit.
La difficulté est que lorsque h devient très petit, le numérateur et le dénominateur deviennent tous les deux très proches de 0. Il est dès lors difficile de savoir ce que devient le quotient.
Est-il possible d'écrire l'expression sous une autre forme, une forme sous laquelle il soit plus facile de trouver ce que devient l'expression au fur et à mesure que h se rapproche de 0 ?
La réponse est oui. C'est possible en utilisant une astuce.

1. Soit a et b deux nombres positifs.
   Écrivez plus simplement l'expression suivante :

   $$\left(\sqrt{a}-\sqrt{b}\right)\left(\sqrt{a}+\sqrt{b}\right) = \_\_\_\_\_$$

2. **(a)** Utilisez ce qui précède pour mettre sous une autre forme le calcul

   $$\frac{\sqrt{3+h}-\sqrt{3}}{h} = \_\_\_\_\_$$

(b) Que devient l'expression au fur et à mesure que h se rapproche de 0 ?

(c) Concluez.

Fort de cette victoire, continuons d'exploiter cette astuce pour en extraire tout ce qu'elle peut nous donner.

3. Adaptez le raisonnement précédent pour calculer la valeur de $\mathrm{d}f/\mathrm{d}x$ en $x = 4$.

Soyons encore plus fort !

4. Adaptez le raisonnement précédent de façon à calculer la valeur de $\mathrm{d}f/\mathrm{d}x$ pour une valeur de x quelconque.

Nous avons donc atteint et même dépassé l'objectif initial du TP 🙂.

DM Les Bonbons 1L – Option Math

Un paquet de bonbons coûte 6 €. Dans chaque paquet, il y a un coupon et avec 3 coupons, on nous offre un paquet de bonbons.

1) Je dispose de 48 €. Vérifier que l'on peut obtenir 11 paquets de bonbons.
2) Je dispose de 96 €. Combien de paquets de bonbons vais-je obtenir ?
3) Je dispose de 200 €. Combien de paquets de bonbons vais-je obtenir ?
4) Ecrire un algorithme qui, à partir de la somme dont je dispose, me renvoie le nombre de paquets de bonbons obtenus.
5) De quelle somme dois-je disposer pour obtenir 30 paquets ?

1) On pose n la somme d'argent disponible. S le nombre de paquets et c le nombre de coupons.

Ici n=48 48=6*8 donc on a 8 paquets : s=8 et 8 coupons : c=8
comme c ≥ 3 on peut avoir des paquets gratuits : 8=3*2+2 on a donc 2 paquets gratuits, donc s= 8+2=10 et c=2+2=4 (les 2 coupons des nouveaux paquets plus les 2 coupons qui nous restaient)
comme c ≥ 3 on peut encore avoir des paquets gratuits 4=3*1+1, on a un paquet en plus donc s=10+1=11 et il nous reste 2 coupons.

Conclusion : avec 48€, j'ai 11 paquets de bonbons.

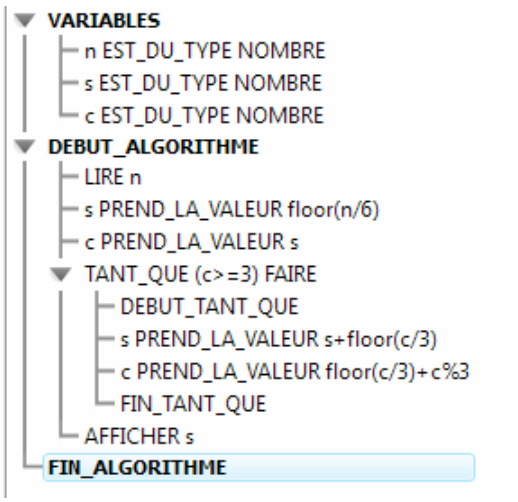
2)3) On obtient 23 paquets avec 96 €et 49 paquets avec 200€.

4) Ecriture de l'algorithme :

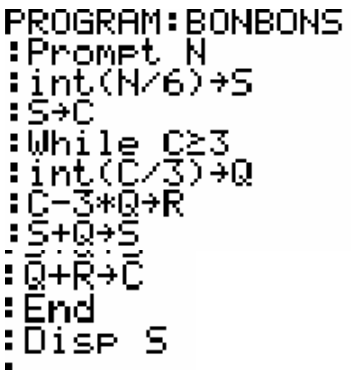
- On demande le nombre n, somme d'argent disponible.
- On divise n par 6, le quotient entier est le nombre de paquets reçus, on appelle s ce nombre, le nombre de coupons c est aussi s.
- Tant que c est supérieur ou égal à 3 on a des paquets gratuits :
  On divise c par 3, le quotient entier est le nombre de paquets reçus que l'on rajoute à s
  Le nombre de coupons est alors égal au nombre de paquets reçus plus les coupons restants
  On recommence le procédé
- On affiche s le nombre de paquets reçus au total

5) On peut programmer cet algorithme :

Cela donne avec Algobox :



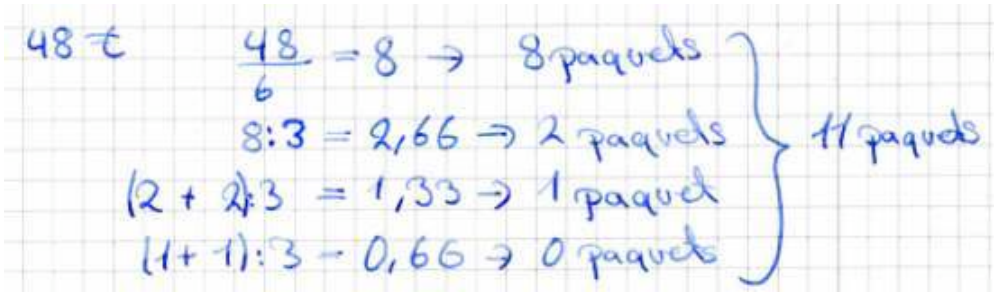
Avec la TI :



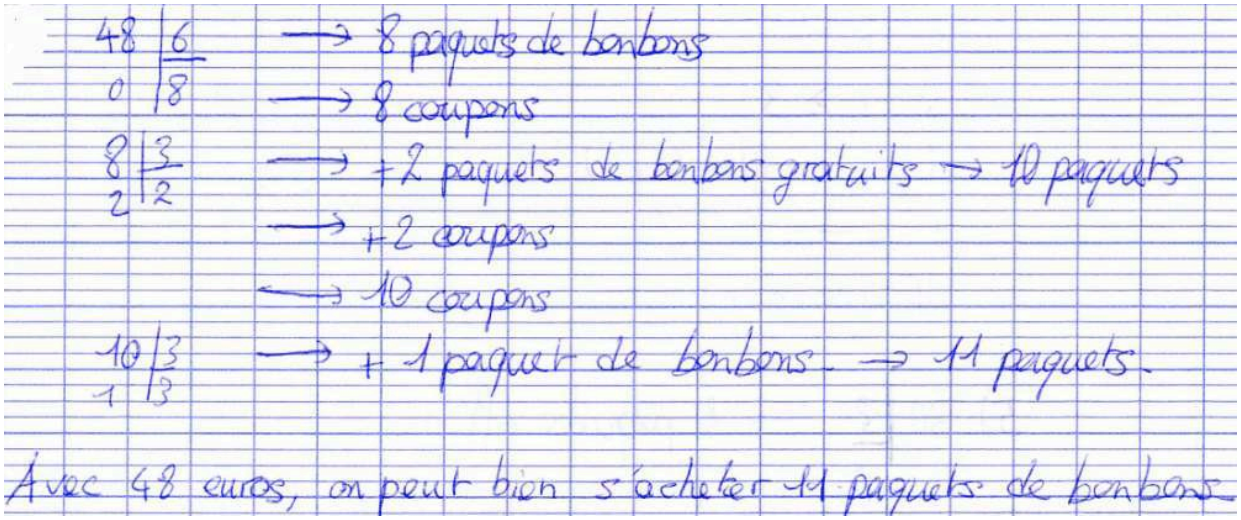
6) En testant quelques valeurs, on trouve qu'il faut 126 €pour avoir 30 paquets. (On a en fait 31 paquets).

Question 1

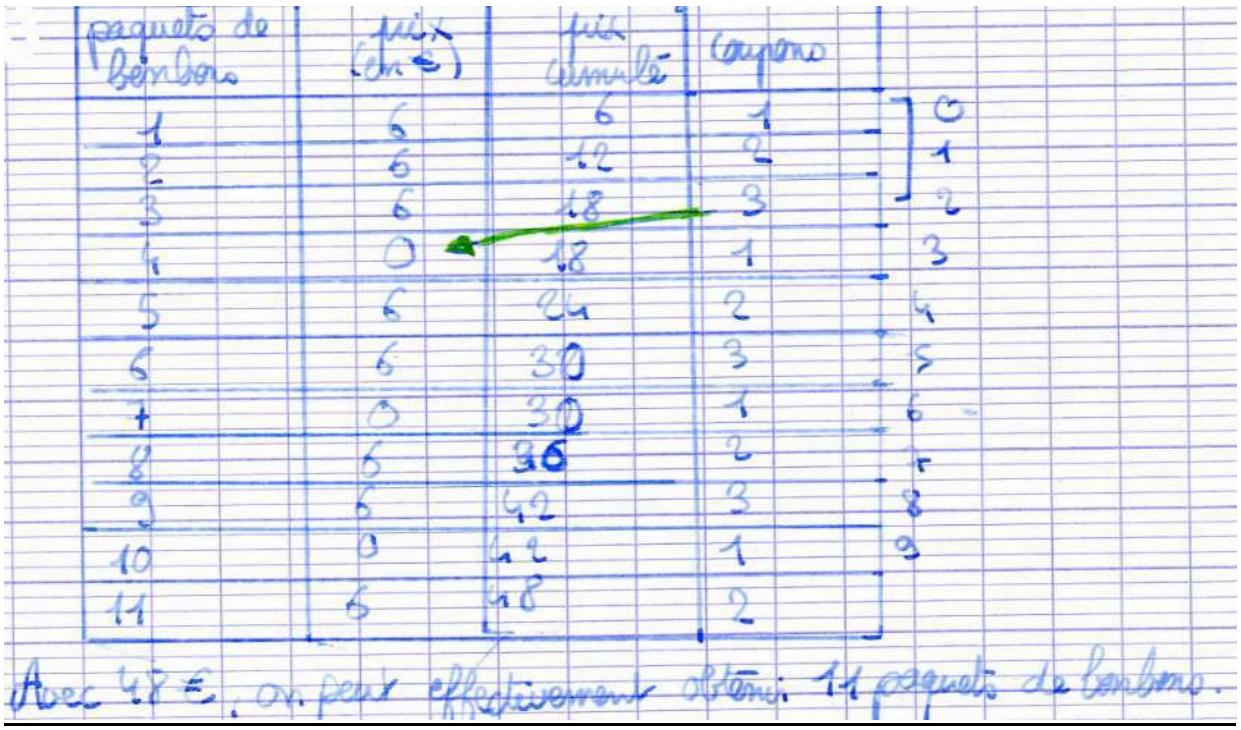
Elève 2



Elève 3

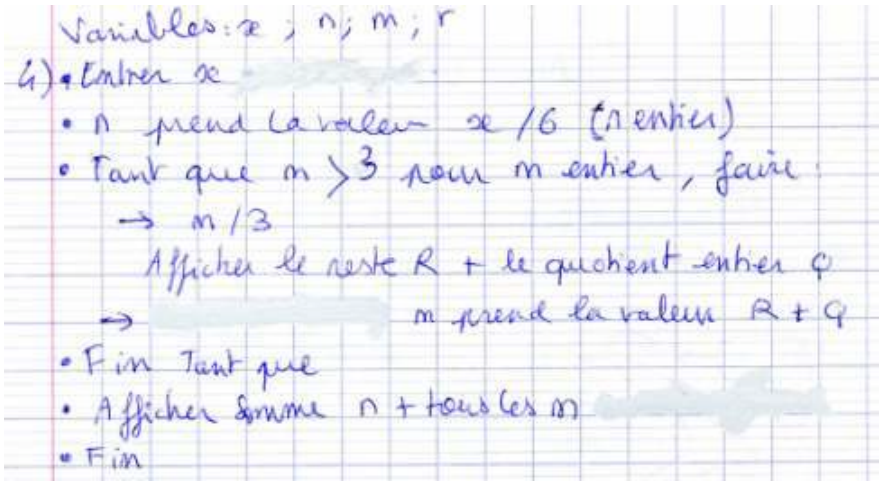


Elève 4

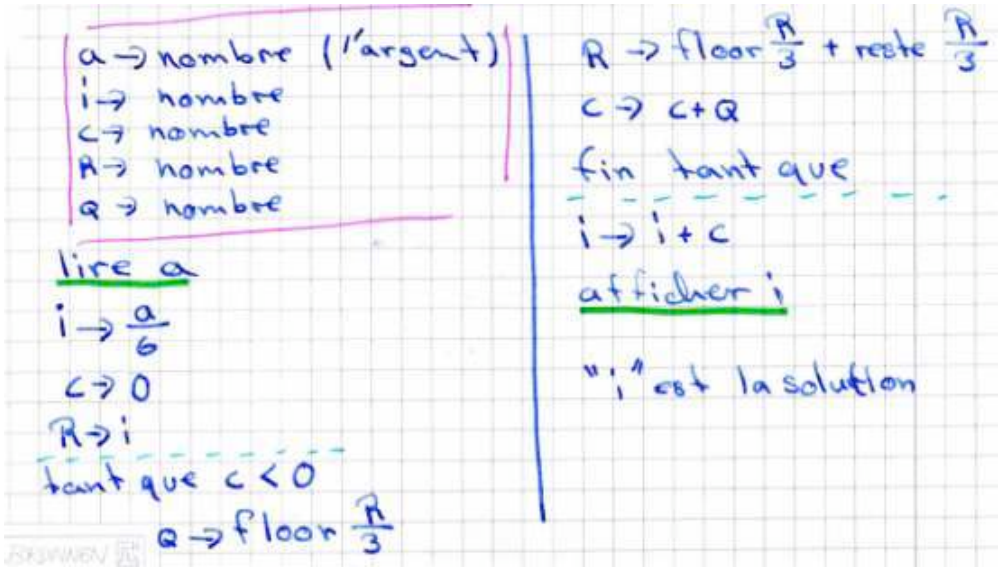


Question 4 : Algorithmes proposés :

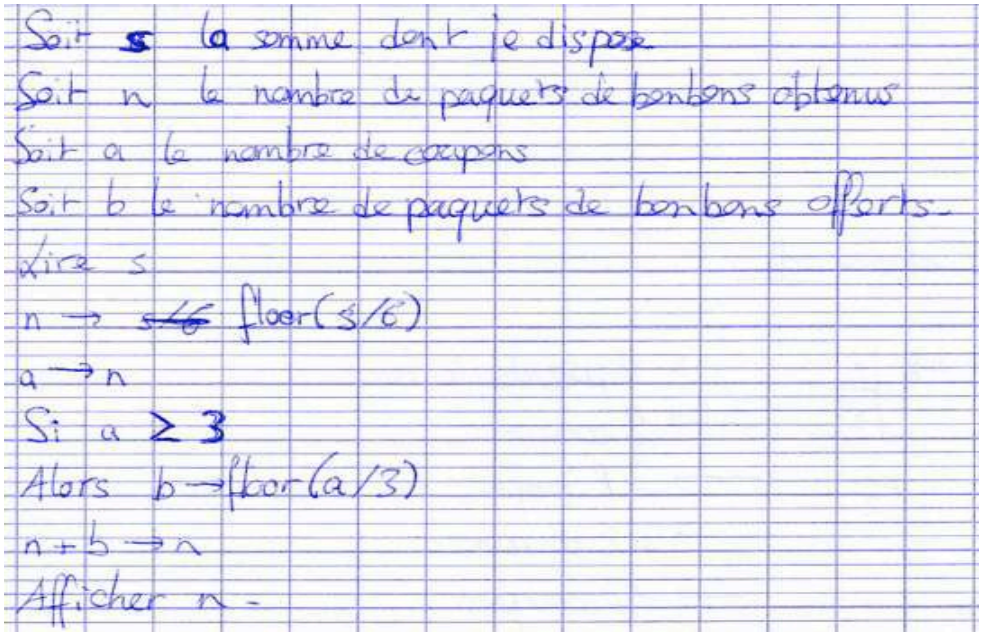
Elève 1



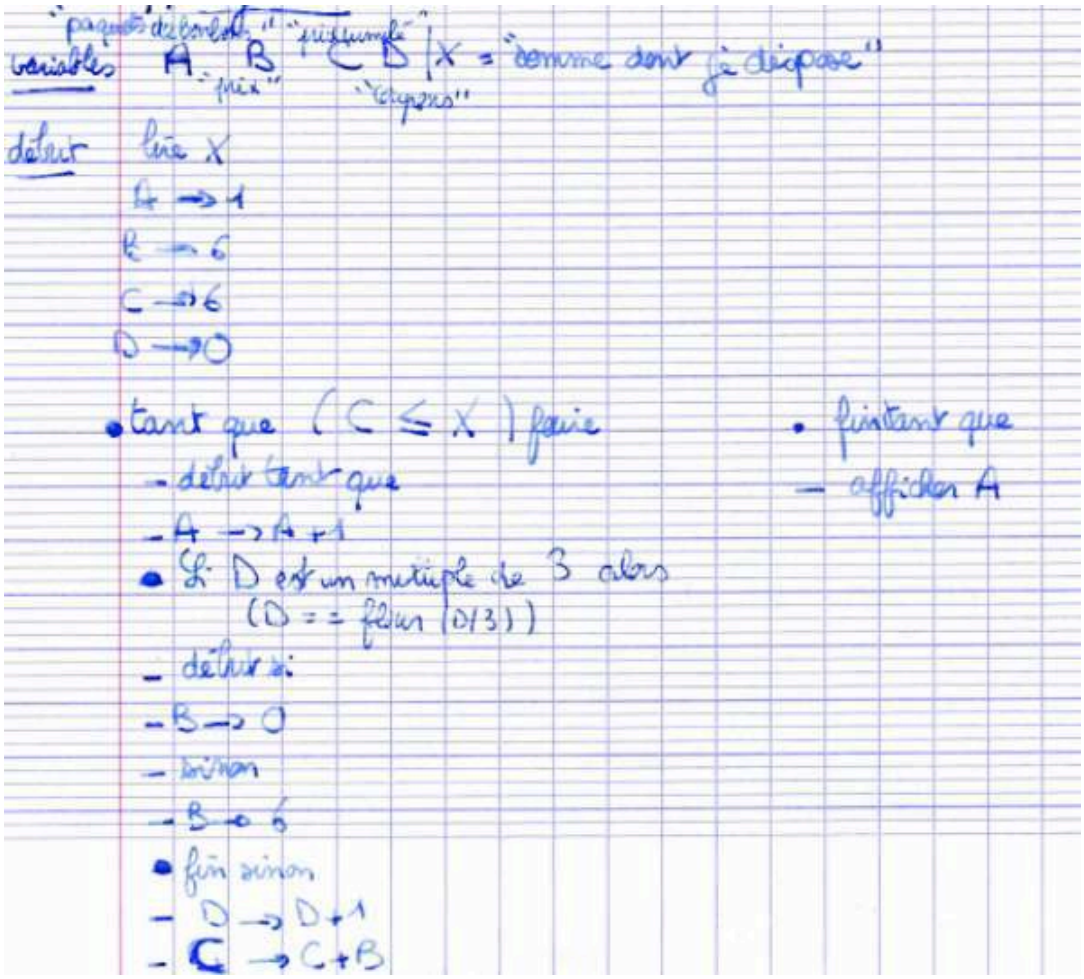
Elève 2



Elève 3



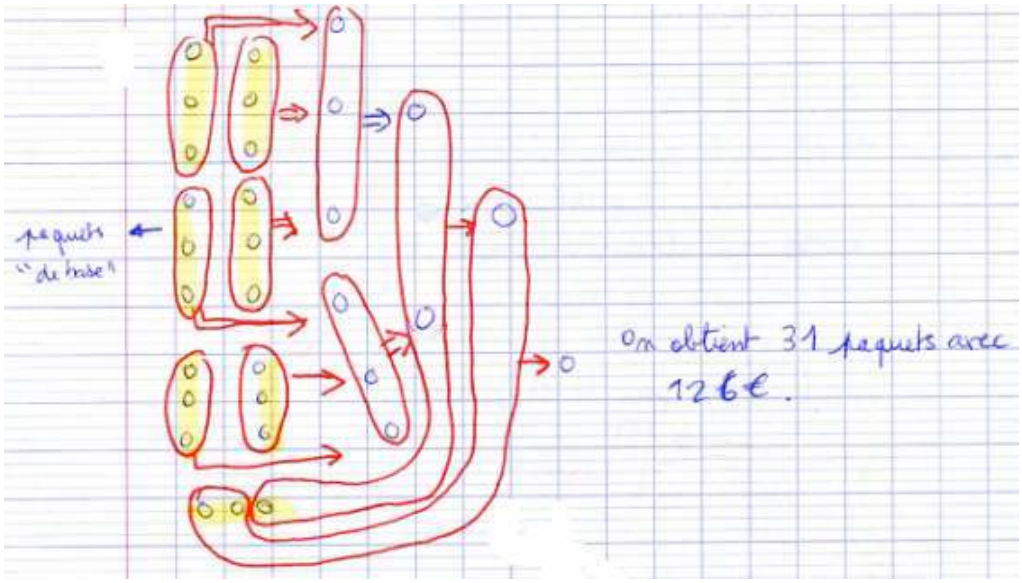
Elève 4



Elève 5

Question 5 :

Elève 1



En Terminale S-Spécialité

- Nombres parfaits – Nombres amicaux *(Scilab)*
- Répartition des nombres premiers – Nombres de Mersenne *(Scilab)*
- Évaluation *(Congruences - Suite de Syracuse - Scilab)*

TS	Spe Math	Arithmétique avec le logiciel SCILAB	TP 1

Scilab est un logiciel gratuit, vous pourrez le télécharger à l'adresse suivante : http://www.scilab.org/download. Une fois Scilab chargé, il faut rajouter le module lycée : dans la barre de menus cliquer sur Applications>Gestionnaire de modules-ATOMS, puis cliquer sur Module lycée, puis sur installer.

I) Quelques exemples d'utilisation

Charger le logiciel ; la page qui apparait s'appelle **la console**. Taper successivement :

*3+6*9*
sqrt(16)
floor(-3.4)
diviseurs(60)
sum(diviseurs(60))

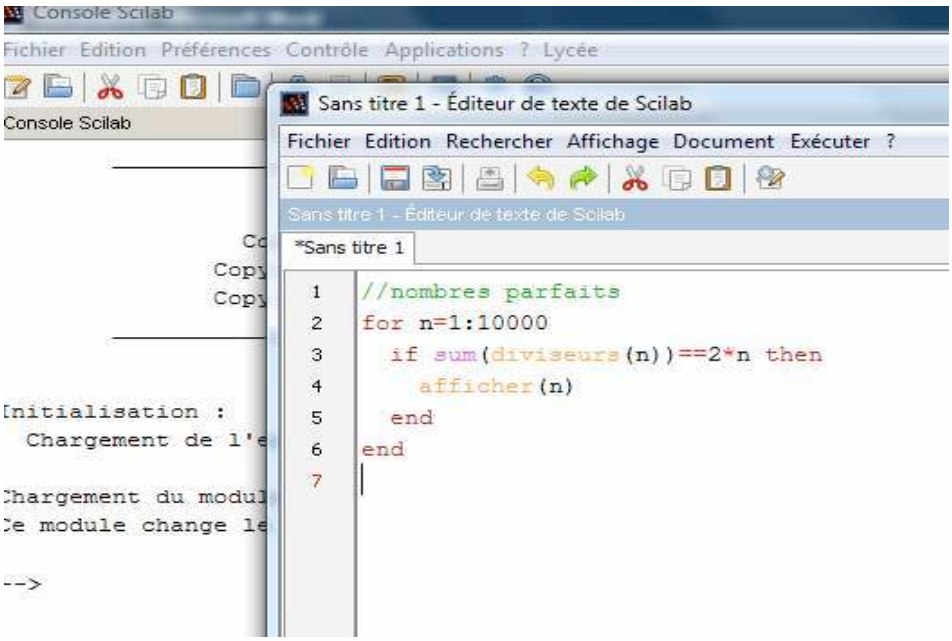
Scilab contient donc des fonctions prédéfinies, mais il est possible de créer ses propres fonctions.

II) Nombres Parfaits – Nombres amicaux

On appelle diviseur strict d'un entier, tout diviseur de cet entier autre que lui-même.
Un nombre parfait est un entier naturel égal à la somme de tous ces diviseurs stricts.
Deux nombres amicaux sont chacun égal à la somme des diviseurs stricts de l'autre.

1) Vérifier que 28 est parfait.
2) Vérifier que 1184 et 1210 sont amicaux.
3) Voici un programme écrit avec le logiciel Scilab pour donner tous les nombres parfaits inférieurs à 10000 :

On écrit le programme dans **l'éditeur** obtenu en cliquant sur l'icône de gauche du menu :



Ctrl+L permet de copier le contenu (ou la sélection) de l'éditeur dans la console. De plus l'exécution est lancée dans la console.
Tester ce programme.

4) Ecrire un algorithme puis le programmer avec Scilab pour donner tous les nombres amicaux inférieurs à 10000.

5) Euclide donne la règle suivante pour trouver des nombres parfaits :

« Si un nombre a s'écrit $2^n(2^{n+1}-1)$ et si le facteur $(2^{n+1}-1)$ est premier, alors a est un nombre parfait. »

a) En utilisant ce théorème, trouver des nombres parfaits.
b) Donner la décomposition de a en facteurs premiers.
c) En déduire la liste des diviseurs de a.
d) Montrer que la somme de ces diviseurs vaut $2a$. Conclure.

Il existe une réciproque de ce théorème due à Euler : « tout nombre parfait pair s'écrit $2^n(2^{n+1}-1)$ avec $(2^{n+1}-1)$ premier. ». Le problème de savoir s'il existe des nombres parfaits impairs n'est toujours pas résolu.

III) **Wanted** A_n *(A partir d'un exercice du manuel Pixel TS spécialité)*

On considère l'algorithme suivant qui calcule les nombres A_n pour n entier entre 1 et 10 :

Pour n allant de 1 à 10
Initialiser A et K en leur donnant la valeur n
Tant que K > 2 :
    Donner à K la valeur K-1
    Donner à A la valeur A*K
Fin_tantque
Donner à A la valeur A -1
Afficher n et A
Fin_pour

1) Programmer sur Scilab cet algorithme.

```
for n=1:10
  a=n;
  k=n;
  while k>2
    k=k-1;a=a*k;
  end
  a=a-1;
  afficher([n,a]);
end
```

2) Donner une expression de A_n en fonction de n.
3) Pour chacune des propositions suivantes, dire si elle est vraie ou fausse en justifiant la réponse:
    a) Il existe des valeurs de n pour lesquelles A_n est premier.
    b) Quelque soit n, A_n est un nombre premier.
    c) Si n est premier alors A_n est un nombre premier.
    d) La réciproque du c) est fausse.
4) Etudier la parité des nombres A_n .

Compte rendu TP1 scilab TS spe Math

TP avec 25 élèves sur 2h en salle informatique avec 2 professeurs

Prérequis : En algorithmique : rien a priori.
En mathématiques : nombres premiers/diviseurs/décomposition en produits de facteurs 1er

Compte rendu de la séance :
1ière heure : I) et II) les élèves travaillent en autonomie et manipulent assez facilement le logiciel.
Nous proposons comme correction :

```
1  //nombres amicaux
2  for n=2:10000
3    s=sum(diviseurs(n))-n;
4    t=sum(diviseurs(s))-s;
5    if t==n then
6      afficher ([n,s])
7    end
8  end
```

Pour la question 5) ils ont besoin d'aide, la décomposition en facteur premiers de *a* n'est pas évidente pour tous. Le calcul de la somme des diviseurs ne parait pas simple, il faut rappeler comment calculer la somme des termes d'une suite géométrique.

2ième heure III)
Il s'agit ici de lire un algorithme. Ils commencent par écrire le programme avec Scilab mais n'en déduisent pas la formule cherchée. Nous proposons alors de lire ensemble l'algorithme par ce que l'on a déjà appelé la mise en scène théâtrale :

3 élèves sont au tableau et le professeur lit l'algorithme, chacun écrivant au fur et à mesure la valeur prise par « sa » variable, voici à peu près le tableau obtenu :

Algorithme écrit	Elève 1 joue le rôle de la variable n	Elève 2 joue le rôle de la variable A	Elève 3 joue le rôle de la variable K	Bilan des entrées et sorties
Pour n allant de 1 à 10	n=1	A=1	K=1	n=1 A=1-1=0
	n=2	A=2	K=2	n=2 A=2-1=1
Initialiser A et K en leur donnant la valeur *n*	n=3	A=3 A=3*2 A=3*2-1	K=3 K=3-1=2	n=3 A=3*2-1=5
Tant que K > 2 : Donner à K la valeur K-1 Donner à A la valeur A*K **Fin_tantque**	----------	A=4 A=4*3 A=4*3*2 A=4*3*2-1	K=4 K=4-1=3 K=3-1=2	n=4 A=4*3*2-1
Donner à A la valeur A -1 Afficher *n* et A **Fin_pour**	n=4 n=5…..ETC			

Les élèves finissent par trouver l'expression de A en fonction de n.
Cette manière de faire permet à chacun de bien comprendre la notion de boucle et la notion de variable.

Scilab est un logiciel gratuit, vous pourrez le télécharger à l'adresse suivante : http://www.scilab.org/download. Une fois Scilab chargé, il faut rajouter le module lycée : dans la barre de menus cliquer sur Applications>Gestionnaire de modules-ATOMS, puis cliquer sur Module lycée, puis sur installer.

I) Quelques exemples d'utilisation

Charger le logiciel ; la page qui apparait s'appelle **la console**. Taper successivement :

*3+6*9*
sqrt(16)
floor(-3.4)
diviseurs(60)
sum(diviseurs(60))
quotient(123,8)
reste(123,8)
factorise(75)
premier(133)
premier(31)
liste_premiers(50)
taille(liste_premiers(50))

Scilab contient donc des fonctions prédéfinies, mais il est possible de créer ses propres fonctions.

II) Répartition des nombres premiers *(A partir d'une activité du manuel Pixel TS spécialité)*

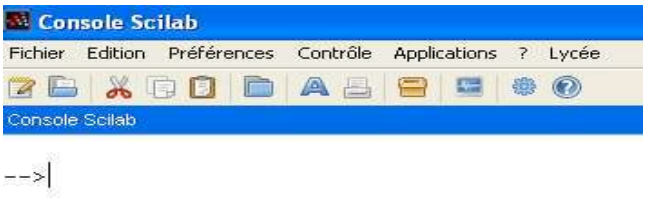
On s'intéresse au nombre $\pi(n)$ de nombres premiers inférieurs ou égaux à un entier naturel n donné.

1) Définition de cette fonction avec Scilab

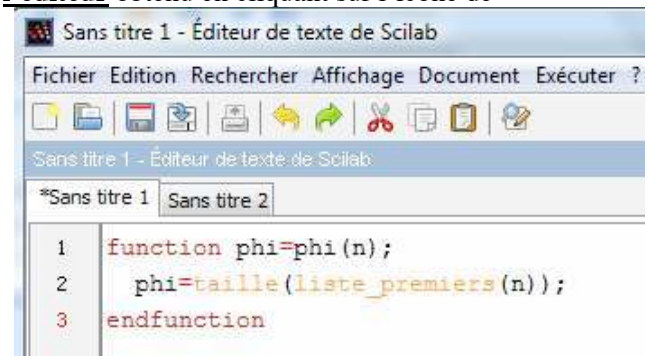
```
function phi = phi(n) ;
  Phi = taille(liste_premiers(n)) ;
endfunction
```

Pour calculer $\pi(50)$:
phi(50)

Lorsque l'on écrit plusieurs lignes dans la console de Scilab il n'est pas facile de corriger en se déplaçant dans le texte. Il est préférable d'écrire les fonctions dans **l'éditeur** obtenu en cliquant sur l'icône de gauche du menu :



Editeur→



Ctrl+L permet de copier le contenu (ou la sélection) de l'éditeur dans la console. De plus l'exécution est lancée dans la console.

On peut tester ce programme :

```
-->phi(50)
 ans  =

    15.
```

2) Densité de ces nombres premiers

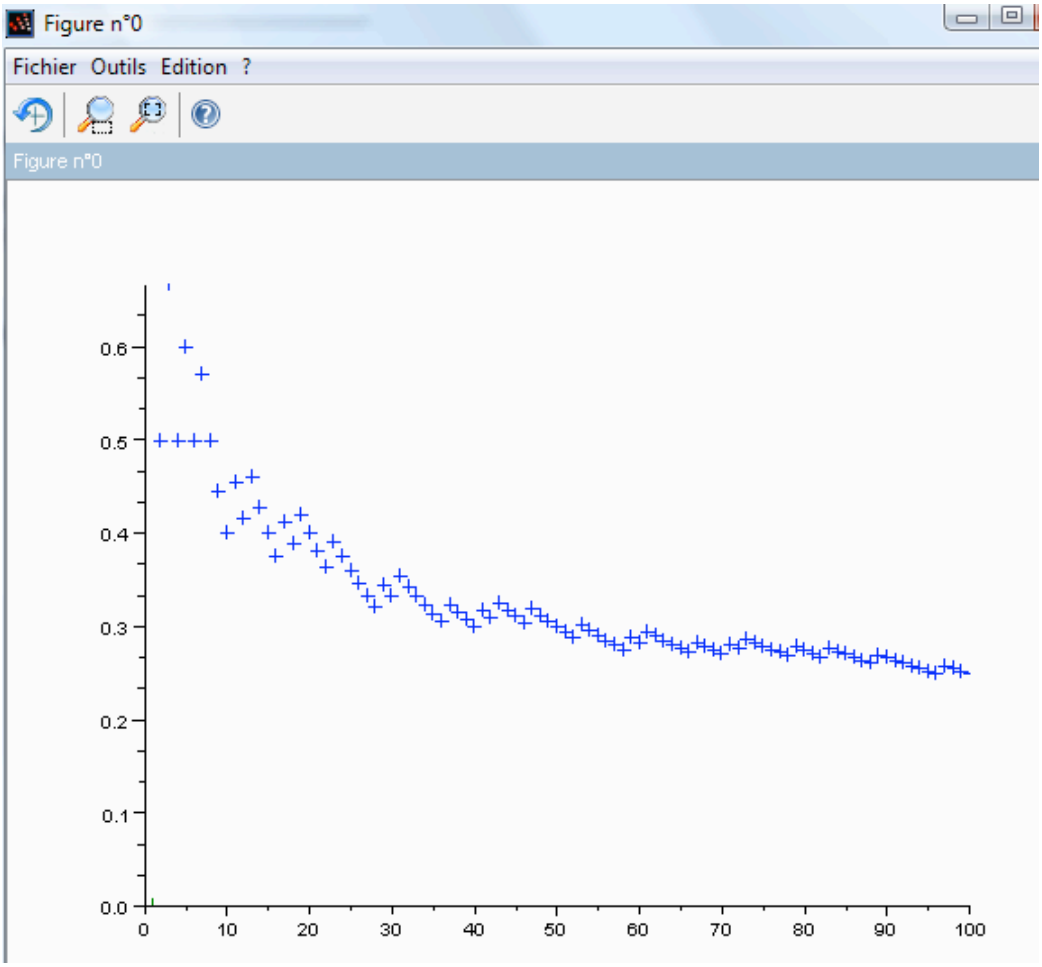
Définir la fonction d telle que $d(n) = \frac{\pi(n)}{n}$.

Tester votre programme pour calculer d(50) et d(100).

Pour représenter graphiquement cette fonction pour $1 \leq n \leq 100$, on écrit :

```
1  for n=1:100
2    plot(n,d(n),"+")
3  end
```

On tape « entrée », on obtient :



Lorsqu'on étudie ainsi une fonction a priori complexe, on cherche, pour la connaitre davantage, à la comparer à des fonctions connues. Vous en connaissez déjà plusieurs fonctions dites de références : les fonctions affines, carrés, inverses….Vous allez cette année en découvrir d'autres : la fonction logarithme et la fonction exponentielle.

Représenter graphiquement cette fonction pour $10^0 \leq n \leq 10^6$

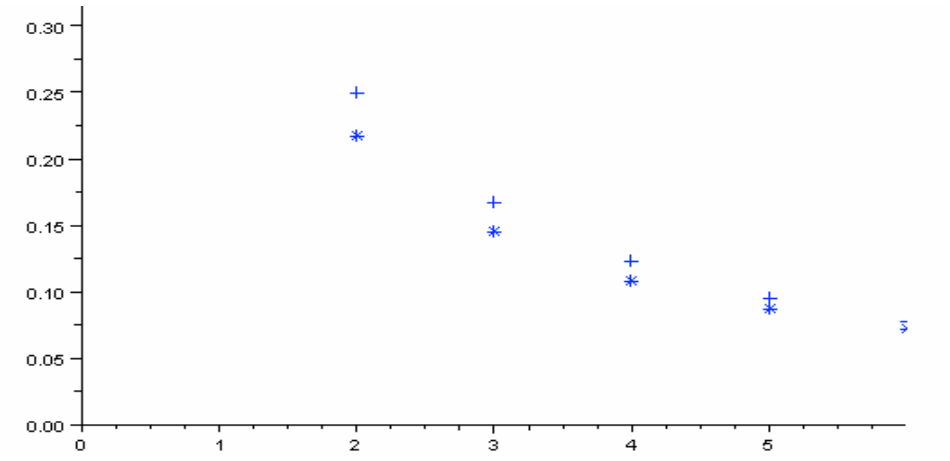
Quel est le problème ?

On décide de remplacer n par log(n) en abscisse. En effet $\log(10^k) = k$.

Faire un graphique des points de coordonnées (log (n) ; d(n)) :

```
for k=0:6
  plot(log10(10^k),d(10^k),"+")
end
```

Déterminer par tâtonnements une valeur de a telle que la représentation graphique soit proche de celle de $x \to \frac{1}{ax}$



La constante cherchée est en fait ln 10. On a donc $d(10^k) \approx \frac{1}{\ln 10 \times k}$. De plus $n = 10^k$ équivaut à $k = \log(n)$ d'où $d(n) \approx \frac{1}{\ln 10 \times \log n}$ c'est-à-dire $d(n) \approx \frac{1}{\ln n}$.

C'est au 18$^{\text{ième}}$ siècle que Gauss aurait démontré que $\frac{\pi(n)\ln(n)}{n} \approx 1$ pour n assez grand. Cette conjecture fut précisée et démontré par Charles de la vallée Poussin et Jacques Hadamard au 19iéme siècle, le résultat étant le suivant : $\lim_{n \to +\infty} \frac{\pi(n)\ln(n)}{n} = 1$.

III) Nombres de Mersenne

Les nombres de Mersenne sont de la forme $n = 2^p - 1$.

1) Supposons que p est premier et p < 20, déterminons les nombres de Mersenne qui sont premiers.

Nous allons écrire un programme avec Scilab, on commence par écrire un algorithme :

Pour p allant de 2 à 20
 Si p est premier alors
  On calcule $n = 2^p - 1$
  Si n est premier alors
  On affiche p et n
  Fin-Si
 Fin-Si
Fin-pour

Traduisons cet algorithme avec Scilab ; On se place dans l'éditeur :

```
1  for p=2:20
2    if premier(p)==%T then
3      n=2^p-1;
4      if premier(n) then afficher ([p,n])
5         end
6    end
7  end
```

Pour lancer le programme, dans l'éditeur taper Ctrl+L

2) Supposons que p n'est pas premier. On peut donc écrire $p = qk$ avec q et k supérieurs ou égaux à 2.
   a) Calculer $\sum_{i=0}^{k-1} (2^q)^i$.
   b) En déduire une factorisation de $2^{qk} - 1$.

3) Montrer que si $2^p - 1$ est premier alors p est premier.

Le 14 novembre 2001, grâce à la mise en commun pendant deux ans et demi du travail de plus de 100000 ordinateurs au sein du GIMPS (Great international Mersenne Prime), a été prouvé que $2^{13466917} - 1$ est premier. (Ce nombre s'écrit avec plus de quatre millions de chiffres.)

Compte rendu TP2 scilab **TS spe Math**

TP 1h avec 16 élèves

Prérequis :
en algorithmique : rien,
en mathématiques : nombres premiers

Compte rendu très rapide de la séance :

I) Ce paragraphe est ici pour rendre le TP2 indépendant du TP1.
II) Pas de problèmes particuliers. Le fait d'avoir à chercher une courbe rapprochant une autre est intéressant.
III) Question 2) et 3) à finir pour la séance suivante.

Evaluation TS SpeMath- En salle informatique avec Scilab -1h

Exercice 1 :

Voici un programme écrit avec le logiciel Scilab, qui donne un tableau des congruences modulo 8 de $3x$ en fonction de la valeur de x modulo 8 :

```
for i=0:7
  afficher ([i,modulo(3*i,8)])
end
```

On obtient :

```
 0.    0.

 1.    3.

 2.    6.

 3.    1.

 4.    4.

 5.    7.

 6.    2.

 7.    5.
```

1) Détailler les opérations qui donnent le chiffre 7 de la deuxième colonne.

---

2) Déduire du tableau la résolution de l'équation $3x \equiv 5(8)$.

---

3) Ecrire un algorithme qui affiche la solution de l'équation $7x \equiv 3(11)$. Programmer cet algorithme avec Scilab. Donner les solutions obtenues.

Appeler le professeur pour qu'il valide votre travail

Exercice 2 :

On considère la suite (U_n) définie par : U_0 est un entier naturel non nul et pour tout entier naturel n, $U_{n+1} = \frac{U_n}{2}$ si U_n est pair et à $3U_n+1$ sinon. Cette suite est appelée suite de Syracuse.

1) Ecrire un programme avec Scilab qui donne les 20 premiers termes de la suite en demandant d'abord la valeur du premier terme. (*On écrira pour cela U = input («1er terme : »)*)

Appeler le professeur pour qu'il valide votre travail.

Utiliser votre programme pour compléter le tableau suivant :

U_0	Les 20 premiers termes de la suite (U_n)
1	1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4
2	2 ; 1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1
3	
4	4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;1
5	5 ;16 ;8 ;4 ;2 ;1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2
6	6 ; 3 ;10 ;5 ;16 ;8 ;4 ;2 ;1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2
7	
8	
9	9 ;28 ;14 ;7 ; 22 ;11 ;34 ;17 ;52 ;26 ;13 ;40 ;20 ;10 ;5 ; 16 ;8 ;4 ;2 ;1
10	10 ; 5 ;16 ;8 ;4 ;2 ;1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4
16	16 ;8 ;4 ;2 ;1 ; 4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1 ;4 ;2 ;1
21	

2) Notons p le plus petit entier tel que $U_p = 1$. Il semblerait que p existe toujours, c'est ce qu'on appelle la conjecture de Syracuse. p est alors appelé la longueur du vol. Le plus grand des p premiers termes de la suite est appelé altitude du vol.

a) Expliquer ce que fait le programme suivant :

```
1  u=input("1er terme:");
2    while u<>1
3      afficher (u);
4      if modulo(u,2)==0 then u=u/2;
5      else u=3*u+1;
6         end
7    end
```


---

b) Améliorer ce programme pour qu'il affiche aussi la longueur du vol.

Appeler le professeur pour qu'il valide votre travail.

---

c) Améliorer encore votre programme pour qu'il affiche aussi l'altitude du vol.

Appeler le professeur pour qu'il valide votre travail.

---

d) Compléter le tableau ci-dessous :

U_0	p	altitude
1		
2		
3		
4		
5		
6	9	16
7		
8		
9		
10		

3) Conjecturer les valeurs de U_0 telles que la suite des valeurs $(U_0, U_1 \ldots\ldots U_p)$ est strictement décroissante. Démontrer cette conjecture.

---

4) a) Déterminer des valeurs de U_0 telles que la suite des valeurs $(U_1 \ldots\ldots U_p)$ est strictement décroissante mais pas la suite $(U_0, U_1 \ldots\ldots U_p)$. Que peut-on dire de ces valeurs ?

---

b) Déterminer les valeurs de *m* telles que $2^m - 1$ soit divisible par 3.

---

c) Justifier que l'on doit avoir $3U_0 + 1 = 2^m$ avec *m* entier naturel strictement supérieur à 2, pour que la suite vérifie les conditions du a).
En déduire cinq valeurs de U_0 qui conviennent.

Bilan évaluation TS : arithmétique avec le logiciel Scilab

Organisation :
Deux groupes de 16 élèves en salle info pendant une heure avec à chaque fois deux enseignants.
Les élèves étaient en parallèle surveillés par une troisième personne pour un devoir « classique ».

Prérequis :
Les élèves ont travaillés les deux TP d'arithmétique avec Scilab donnés dans la brochure.
Le cours sur les congruences est terminé.

Remarques générales :
Nous avons rencontré beaucoup de problèmes dus à la syntaxe à utiliser pour écrire les programmes demandés. Il a été difficile d'intervenir « également » pour chaque élève ; et il est donc assez difficile d'évaluer correctement le travail effectué par chacun. Les élèves ont trouvé cela assez difficile.

Barème :
Nous avons décidé de mettre une note globale sur 4 points en notant le devoir fait en parallèle sur 16 points. Les 4 points sont répartis de manière suivante : 1 point pour l'ex1, 1 point pour le 1) de l'ex2, 1 point pour le 2) a)de l'ex2 et 1 point pour le 1)b et c de l'ex2.
Le reste de l'épreuve a été réalisée par très peu d'élèves.

Erreurs d'élèves :
En dehors des très nombreuses erreurs de syntaxes (, à la place de ; -espace en trop- ; manquant-pas de then s'il n'y a pas de else…etc.) :

Ex 1 -3) :
Les élèves ont sans problème fait afficher un tableau semblable à celui de la question 1) mais n'ont pas compris qu'il s'agissait de faire afficher uniquement les solutions.

Pour calculer la valeur b de a modulo n, ils sont nombreux à écrire b=([b,modulo(a,n)]) plutôt que b=modulo(a,n).

D'autres n'ont pas fait afficher le contenu de la « bonne » variable.

Ex2-1) Pour tester la parité d'un nombre n, certains ont écrit if n=pair(n) ou encore if n pair.
On leur a rapidement dit qu'ils pouvaient utiliser le programme de la question suivante pour éviter trop de problèmes de syntaxe.
Ici, ils n'ont pas tous pensé à une boucle « for ». Certains ont écrit « while u>0 » ou « while u<>0 » par exemple.
D'autres ont écrit un programme qui nous semblait correct mais qui ne fonctionnait pas !! Nous n'avons donc pas toujours trouvé les erreurs.

Ex2-2) Pas de problème particulier. La condition d'arrêt du programme n'a cependant pas été écrite par tous.
L'idée d'introduire une variable pour compter n'est pas naturelle.

La suite a été traité par peu d'élève. Un élève a fini.

Améliorations à faire :
Ex1-3) pour légitimer le fait d'afficher les solutions plutôt que le tableau de congruences, on peut plutôt proposer de résoudre une équation du type $7x \equiv 3 \bmod 189$.

Ex2 Il faudrait proposer d'abord de lire le programme donnant les 20 premiers termes de la suite et demander ensuite de le transformer pour obtenir les termes jusqu'à l'obtention de 1.

Conclusion :
Nous ne recommencerons pas cette évaluation à cause des trop nombreux problèmes de syntaxe. Le sujet peut par contre, une fois modifié, servir de TP.

Correction partie algorithmique :

Ex1

```
for i=0:10
  if modulo(7*i,11)==3 then
    afficher (i)
  end
end
```

Ex2

1)

```
u=input("1er terme:");
  for n=1:20
    afficher (u);
    if modulo(u,2)==0 then u=u/2;
    else u=3*u+1;
       end
  end
```

Autre version (on peut ici faire un graphique) :

```
u(1)=input("1er terme:");
  for n=1:20
    afficher ([n,u(n)]);
    if modulo(u(n),2)==0 then u(n+1)=u(n)/2;
    else u(n+1)=3*u(n)+1;
       end
  end
  clf;plot(u,"*r")
```

2) a)

```
U0= input("U0=")
U=U0
p=1
  while U<>1
    afficher (U)
    if modulo(U,2)==0 then U=U/2;
    else U=3*U+1;
    end
   p=p+1 ;
end
afficher (p)
```

b)

```
U0= input("U0=")
U=U0
p=1
a=U0
  while U<>1
    afficher (U)
    if modulo(U,2)==0 then U=U/2;
    else U=3*U+1;
    end
    p=p+1 ;
    a=max(a,U);
end
afficher (p);
afficher(a)
```

Annexe : La machine humaine

L'ordinateur humain

Les éléments de la machine

Comme tout ordinateur, l*ordinateur humain* **Hominum 2011** exécute des programmes. Pour fonctionner, l'**Hominum 2011** nécessite la participation de plusieurs personnes. Chacune de ces personnes simule un élément précis de la machine selon un mode opératoire précis qui ne demande aucune initiative, autrement dit à un algorithme.
Comme le montre l'annexe *A* cette machine est constituée :

- du *lecteur de programme* qui contient les instructions à exécuter ;
- de la *mémoire* où l'on peut stocker des nombres ;
- du *processeur* qui est le chef d'orchestre de la machine. Il envoie des commandes aux autres composants de la machines et échange des informations avec eux. ;
- de l'*unité de calcul* qui évalue les expressions logiques et arithmétiques ;
- de l'*afficheur* qui gère de l'écran.

Cinq personnes sont donc nécessaire pour faire fonctionner cet ordinateur.
Une sixième personne est bien sûr nécessaire : l'*uitilisateur* de l'ordinateur.

La mémoire

La *mémoire* est constituée d'un maximum de 26 cases, identifiées chacune par une lettre de l'alphabet et pouvant mémoriser un nombre entier positif ou négatif. Ce nombre entier ne peut avoir qu'au maximum 6 chiffres.
Une case mémoire est appelée une *variable*.
Le gestionnaire de la mémoire peut recevoir les ordres suivants :

- **« Écris le nombre <N> dans la variable <V> »**
  Si <**V**> n'est pas une lettre de l'alphabet ou si le nombre <**N**> n'est pas un entier ou s'il comporte plus de 6 chiffres le gestion de mémoire répond **« Erreur »** .
  Le gestionnaire de mémoire crée la variable <**V**> si elle n'existe pas, et mémorise le nombre <**N**> dans cette variable.
- **« Lis la variable <V> »**
  Si <**V**> ne correspond à aucune variable en mémoire, le gestionnaire répond **« Erreur »** , sinon il communique le nombre mémorisé dans la variable.
- **« Efface la mémoire »** : toutes les variables sont supprimées.

Le lecteur de programme

Le *programme* est constitué de lignes numérotées et appelées *instructions*. Une instruction est un texte destiné à être lu par le *processeur* et qui déclenche des actions particulières de sa part.
Le *processeur* tient à jour un *compteur d'instructions* (*CI*) qui est visible par le *lecteur du programme*. Chaque fois que le processeur prononce le mot **« Instruction »** le *lecteur du programme* écrit dans le *registre d'instruction* (*RI*) la ligne

numéro *CI* du programme. Si cette ligne n'existe pas le *lecteur du programme* répond **« Erreur »** .

▹ L'afficheur

L'*afficheur* est l'élément de la machine qui s'occupe de l'affichage sur l'écran de l'ordinateur.

L'*afficheur* peut recevoir les ordres suivants :

- **« Efface l'écran »**
- **« Écris : <TEXTE> »**
  L'afficheur écrit <TEXTE> en commençant au début d'une nouvelle ligne.

▹ L'utilisateur

L'*utilisateur* lance l'exécution du programme chargé en mémoire en donnant l'ordre **« Démarre »** au *processeur*.

L'*utilisateur* interagit avec l'ordinateur par l'intermédiaire du clavier. Le texte saisi par l'*utilisateur* est placé dans le *registre-clavier* (*RC*) du processeur. L'*utilisateur* ne peut écrire un texte dans *RC* que si ce registre est vide.

Le processeur s'attend toujours à lire dans *RC* un nombre entier positif ou négatif ayant au maximum 6 chiffres. Si ce n'est pas le cas, l'exécution est interrompue et un message d'erreur est affiché.

▹ L'unité de calcul

L'*unité de calcul* est la partie du *processeur* qui est chargé d'évaluer les expressions arithmétiques et logiques.

Une expression est constituée :

- de nombres entiers positifs ou négatifs ayant au maximum 6 chiffres ;
- d'opérateurs arithmétiques ($+ - \times$ /%) ;
- d'opérateurs de comparaison ($= \neq < > \leqslant \geqslant$) ;
- de parenthèses.

L'opérateur / calcule le quotient entier de la division. L'opérateur % calcule le reste de cette division.

La valeur de l'expression est :

- soit un entier positif ou négatif (sur 6 chiffres au maximum) s'il s'agit d'une expression arithmétique ;
- soit l'une des valeurs "vrai" ou "faux" s'il s'agit d'une expression logique.

Le tableau ci-dessous donne la valeur de quelques expressions :

Expression	$5+3\times4$	$(5+3)\times4$	$26/3$	26 % 3	$2\leqslant6$	$4>7$
Valeur	17	32	8	2	vrai	faux

Lorsqu'une expression apparait dans le registre *expression* (*RE*) l'*unité de calcul* évalue cette expression et place le résultat dans le registre *valeur* (*RV*).

Si l'expression est incorrecte l'*unité de calcul* répond **« Erreur »** .

Si l'expression provoque une division par zéro l'*unité de calcul* répond **« Division par zéro »** .

Si l'expression donne un nombre avec plus de six chiffres, l'*unité de calcul* répond **« Dépassement de capacité »** .

▹ Le processeur

Le *processeur* est le « chef d'orchestre » de la machine. C'est lui qui décode chacune des instructions du programme et qui commande tous les autres composants afin d'accomplir les actions associées à cette instruction.

```
1:  Attendre l'ordre de démarrage
2:  Commander l'effacement de la mémoire
3:  Commander l'effacement de l'écran
4:  Mettre la valeur 1 dans CI
5:  Commander la lecture de l'instruction
6:  Tant que l'instruction est différente de STOP
7:      Ajouter 1 à CI
8:      Si c'est une instruction ECRIS
9:          Lire les variables éventuellement présentes dans l'expression
10:         Placer l'expression dans RE
11:         Commander l'évaluation de l'expression
12:         Stocker le contenu de RV dans la variable
13:     sinon si c'est une instruction DEMANDE
14:         Attendre la présence d'un texte dans RC
15:         Lire le texte et le supprimer de RC
16:         Stocker le contenu de RC dans la variable
17:     sinon si c'est une instruction AFFICHE
18:         Lire les variables éventuellement présentes dans l'instruction
19:         Commander l'affichage du texte
20:     sinon si c'est une instruction VAS
21:         Mettre dans CI la valeur indiquée par l'instruction
22:     sinon si c'est une instruction TESTE
23:         Lire les variables éventuellement présentes dans l'instruction
24:         Placer l'expression dans RE
25:         Commander l'évaluation de l'expression
26:         Si RV contient "vrai"
27:             Ajouter 1 à CI
28:         fin du si
29:     fin du si
30:     Commander la lecture de l'instruction
31: Fin du tant que
```

De nombreuses situations d'erreurs peuvent se produire. Lorsqu'une erreur survient le *processeur* s'arrête et attend un nouvel ordre de démarrage.

❑ Le langage

Le langage **Hominum 2011** comporte 6 instructions différentes.

Instruction	Effet
ECRIS	Stocke la valeur d'une expression dans la mémoire
DEMANDE	Lis un nombre au clavier et le stocke dans la mémoire
AFFICHE	Affiche un texte ou une variable à l'écran
VAS	Indique la prochaine instruction à exécuter
TESTE	Saute une instruction si l'expression est vraie
STOP	Stoppe l'exécution du programme

Instruction	Syntaxe	Exemple
ECRIS	ECRIS {Expression} → {Variable}	ECRIS $A+3\times B \rightarrow C$
DEMANDE	DEMANDE {Variable}	DEMANDE A
AFFICHE	AFFICHE {Texte ou variable}	AFFICHE "Bonjour"
VAS	VAS {Ligne}	VAS 7
TESTE	TESTE {Expression}	TESTE $A \leqslant B$
STOP	STOP	STOP

Voici des exemples de programme écris dans le langage **Hominum 2011**.

Programme n°1

```
1:  AFFICHE "Nombre ?"
2:  DEMANDE N
3:  AFFICHE N
4:  ECRIS 0 → S
5:  ECRIS 0 → K
6:  TESTE K ≠ N
7:  VAS 12
8:  ECRIS K + 1 → K
9:  ECRIS S + K → S
10: AFFICHE S
11: VAS 6
12: AFFICHE "RÉSULTAT : ", S
13: STOP
```

Programme n°2

```
1:  AFFICHE "Premier nombre ?"
2:  DEMANDE A
3:  AFFICHE A
4:  AFFICHE "Deuxième nombre ?"
5:  DEMANDE B
6:  AFFICHE B
7:  TESTE A % B ≠ 0
8:  VAS 14
9:  ECRIS A % B → R
10: ECRIS B → A
11: ECRIS R → B
12: AFFICHE A, " ", B
13: VAS 7
14: AFFICHE "RÉSULTAT : ", B
15: STOP
```

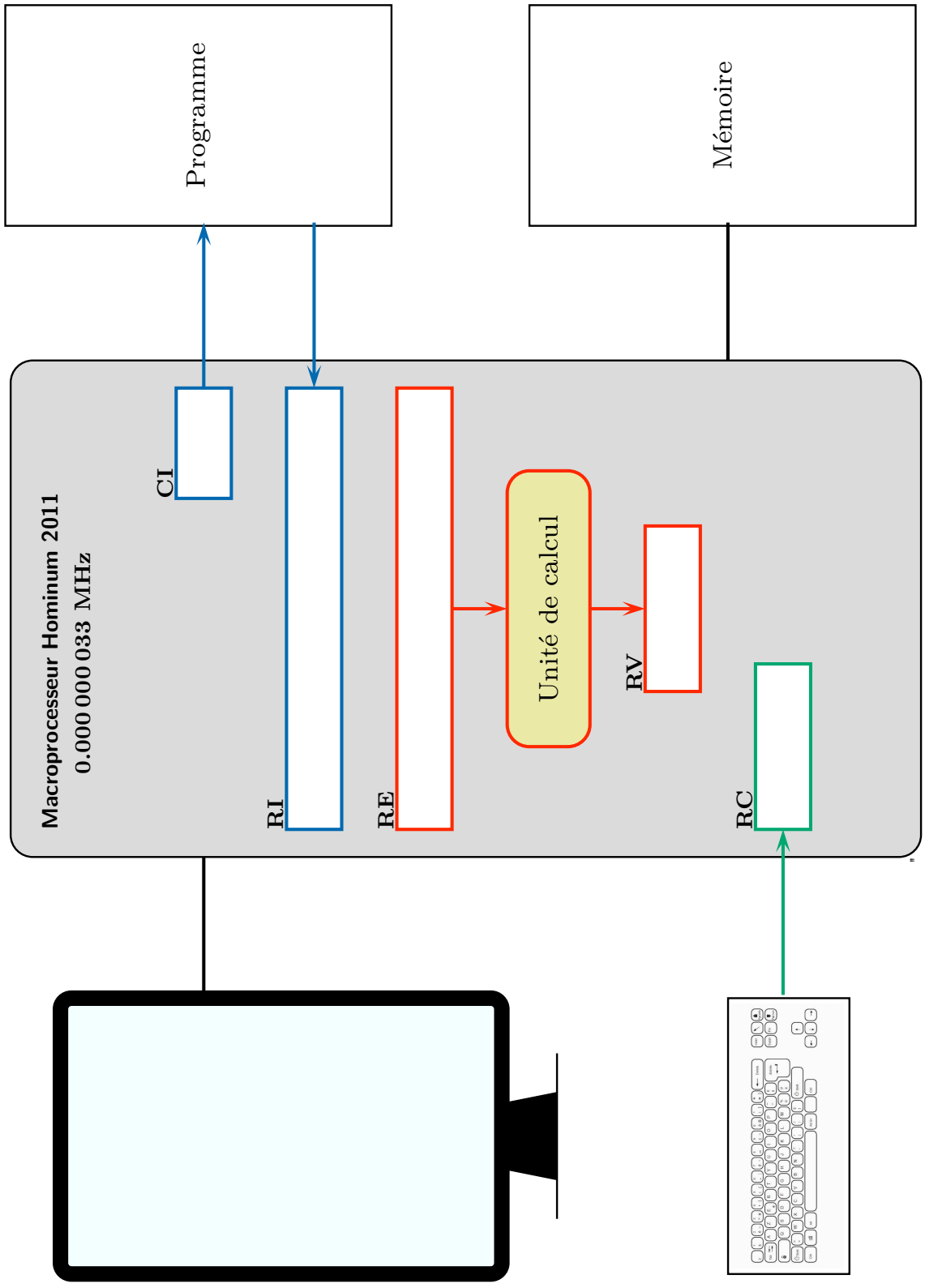
❑ Tests des éléments de la machine

▹ La mémoire

- Efface la mémoire
- Écris 51 dans A
- Écris 32 dans B
- Lis $A+B$
- Lis A
- Lis C
- Écris $2+2$ dans B
- Lis B
- Écris 1 234 567 dans A
- Lis A
- Écris MAX
- Lis MAX
- Écris 3 dans A
- Lis A
- Efface la mémoire
- Lis A

▹ L'unité de calcul

- $3+5\times 2$
- $6/((2\times 4)-8)$
- $29/3$
- 29 % 3
- $555\,444\times 10$
- $3+5)\times 2$
- $24\times 3\leqslant 6\times 11$


Programme
Mémoire
Macroprocesseur Hominum 2011
0.000 000 033 MHz
CI
RI
RE
Unité de calcul
RV
RC

FICHE SIGNALÉTIQUE

Introduction de l'algorithmique au lycée

Auteurs : BALLIOT Anne, DARRIGRAND Éric, GAUDE Laurence, MEUNIER Michèle, MILLET Michèle, PINSARD Denis.

Éditeur : IREM de Rennes.

Date : Décembre 2011

Niveau : Lycée – Seconde.

Mots clés :

Algorithmique, approche graduée d'un algorithme, programmation, outils informatiques, mise en scène théâtrale, fiches d'activité.

Résumé :

Cette brochure présente les travaux d'un groupe de recherche-formation qui s'est intéressé à l'enseignement de l'algorithmique au lycée, principalement en Seconde.
Une première partie présente les réflexions menées par le groupe sur les sujets suivants : approche graduée d'un algorithme ; objectifs envisageables avec une classe de Seconde ; les différents moyens informatiques et une alternative : la mise en scène théâtrale ; les difficultés rencontrées par les élèves ou l'enseignant.
Dans une deuxième partie, nous proposons des activités testées en classes de Seconde, mais aussi en classes de Première et Terminale.

Les activités sont aussi disponibles pour la plupart sur le site web du groupe : http://www.irem.univ-rennes1.fr/recherches/groupes/groupe_algo/index.htm

Format	Nombre de pages	Prix	Tirage
21 x 29,7	**150**	**12 €**	**200 ex.**

ISBN 2-85728-075-0

I.R.E.M de RENNES – Université de RENNES 1
Campus de Beaulieu
35042 RENNES CEDEX
mél : secirem@univ-rennes1.fr
Site WEB : http://www.irem.univ-rennes1.fr