Algorithmes (4)
Boucles « Pour »

I. Exemple 1

1°) Situation

Les parents de Léa versent 50 € sur un livret à sa naissance.

Ils versent ensuite 15 € sur ce livret le premier mois, puis 16 € le deuxième mois, puis 17 € le troisième mois…
Chaque mois le versement mensuel augmente donc de 1 €.

On désire rédiger un algorithme donnant en sortie la somme en euros versée sur le livret de Léa au bout de 3 ans (36 mois).

2°) Analyse du problème

Il s'agit d'un calcul itératif. On va mettre en place une procédure itérative.
On a un processus identique qui se répète un certain nombre de fois.
L'algorithme comporte 36 étapes (répétitives).
On va utiliser une structure algorithmique appelée « boucle ».
On va dire à l'algorithme de faire (d'exécuter) 36 fois la même instruction.

3°) Proposition d'algorithme (rédigé en langage naturel)

Initialisations :
S prend la valeur 50
M prend la valeur 15

Traitement :
S prend la valeur $S+M$
M prend la valeur $M+1$

S prend la valeur $S+M$
M prend la valeur $M+1$

S prend la valeur $S+M$
M prend la valeur $M+1$

S prend la valeur $S+M$
M prend la valeur $M+1$

} 36 fois

Sortie :
Afficher la somme S

On peut dire ce que représentent les variables S et M.
S représente la somme disponible sur le livre de Léa (en euros).
M représente la somme versée par les parents sur le livret (en euros)

On devrait écrire 36 fois le même bloc d'instructions.
Chaque fois la valeur de S s'efface et prend une nouvelle valeur ; de même pour la variable M.

Le dernier bloc d'instruction pourrait se réduire à «S prend la valeur $S+M$ ». En effet, l'instruction « M prend la valeur $M+1$ » ne sert à rien puisque l'on s'arrête après.

Pour simplifier l'écriture de l'algorithme, on va utiliser une boucle « Pour ».
Pour cela, on rédige une instruction, dans un seul bloc, qui sera répétée 36 fois (c'est le principe même des boucles, qu'il faut bien avoir compris dans ce chapitre).

Initialisations :
S prend la valeur 50
M prend la valeur 15

Traitement :
Pour *i* variant de 1 à 36 **Faire**
    S prend la valeur $S+M$
    M prend la valeur $M+1$
FinPour

Sortie :
Afficher la somme S

Cet algorithme comporte 3 étapes : initialisations, traitement, sortie. Il n'y a pas d'entrée.

Cet algorithme fait intervenir 3 variables (S, M, *i*). On pourrait les écrire dans une partie de déclaration des variables (ces trois variables sont des entiers).

La variable S représente le montant en euros de la somme sur le livret.
La lettre S désigne une variable* dont le contenu va évoluer au fur et à mesure de l'algorithme.
Donc dire que S prend la valeur $S+M$ n'a rien d'absurde.
On ne dit pas que $S=S+M$; on n'utilise pas non plus la lettre S′ en disant que $S'=S+M$.

* Une variable est comme une boîte dans laquelle on peut « stocker » une valeur.

La variable M représente le montant du versement en euros effectué par les parents de Léa.

Ces deux variables sont initialisées c'est-à-dire qu'on leur assigne des valeurs au départ.

La variable *i* joue un rôle un peu particulier. C'est une variable interne ou locale. Elle augmente de 1 à chaque fois (pas de 1). La variable *i* est automatiquement incrémentée de 1 à chaque fois.
i n'a pas de rôle dans un calcul. Elle compte le nombre d'itérations (*i* comme « itération »).

Elle n'intervient ni avant, ni après dans l'algorithme.

Commentaires sur cet algorithme :

1. L'algorithme est composé d'une boucle « Pour » à l'intérieur de laquelle interviennent deux instructions itératives : « S prend la valeur $S+M$ » et « M prend la valeur $M+1$ ».

2. Ces deux instructions se réfèrent à l'énoncé.

3. Ces deux instructions qui interviennent dans la boucle ne sont pas interchangeables car M intervient dans la deuxième instruction.

Comme pour tout algorithme, le contenu des variables va évoluer au fur et à mesure du déroulement de l'algorithme.

4°) Fonctionnement de l'algorithme et réponse au problème posé

En faisant fonctionner l'algorithme « à la main » ce qui est très long ou mieux, en le programmant, on obtient la réponse au problème posé.

Voici un extrait du tableau d'évolution des variables.

	Valeurs initiales	1ère étape	2e étape	3e étape		35e étape	36e étape
i		1	2	3			36
M	15	16	17	18		50	51
S	50	66	83	101		1169	1220

Pour comprendre comment fonctionne la boucle, il faut se référer à la notion de variable (qui garde en mémoire une valeur).

1ère étape :
La variable *i* prend la valeur 1.
La variable S prend la valeur 16
La variable M prend la valeur 66.

2e étape :
La variable *i* prend la nouvelle valeur 2 (l'ancienne valeur est effacée de la mémoire)
La variable S prend la valeur 17 (l'ancienne valeur est effacée).
La variable M prend la valeur 83 (l'ancienne valeur est effacée).

Et ainsi de suite jusqu'à la valeur 36 pour *i*.

Au bout de 3 ans, Léa aura 1 220 €.

II. Exemple 2

1°) Situation

Même situation qu'au **I**.

On désire rédiger un algorithme donnant la somme versée sur le livret de Léa au bout d'un certain nombre de mois N.

2°) Analyse du problème

On doit refaire le même type d'algorithme à la différence qu'il faudra rajouter une étape d'entrée permettant de saisir le nombre N de mois.

3°) Proposition d'algorithme (rédigé en langage naturel)

Entrée :
Saisir le nombre N de mois

Initialisations :
S prend la valeur 50
M prend la valeur 15

Traitement :
Pour *i* variant de 1 à N **Faire**
| S prend la valeur $S+M$
| M prend la valeur $M+1$
FinPour

Sortie :
Afficher la somme S obtenue au bout de N mois

L'algorithme fait intervenir 4 variables (N, S, M, *i*).
La variable *i* n'intervient pas dans le calcul.

Cet algorithme comporte 4 étapes : entrée, initialisations, traitement, sortie.

4°) Fonctionnement de l'algorithme

On se propose de faire fonctionner l'algorithme précédent « à la main » pour $N = 4$ à l'aide d'un tableau (tableau de suivi des variables), indiquant les valeurs des variables S et M prises durant les différentes étapes de la boucle.

On considère $N = 4$.
i prend donc successivement les valeurs 1 (début de la boucle), 2, 3, 4 (fin de la boucle).
i est la variable de boucle.

Le tableau suivant montre l'évolution des variables S et M à chaque itération (ou à chaque « passage » dans la boucle).

	S	M
Initialisation	50	15
Fin de la première itération (pour $i=1$)	65	16
Fin de la deuxième itération (pour $i=2$)	81	17
Fin de la troisième itération (pour $i=3$)	98	18
Fin de la quatrième itération (pour $i=4$)	116	19

La somme versée sur le livret de Léa au bout de 4 mois est donc égale à **116 €**.

La valeur 19 portée dans la dernière ligne de la colonne M serait la valeur du montant que les parents verseraient sur le livret de Léa le cinquième mois.

III. Syntaxe de l'instruction itérative ou boucle « Pour » (boucle avec compteur)

L'instruction itérative **Pour *i* variant de 1 à N Faire *Instructions* FinPour** est une **boucle**.
Une boucle permet de répéter plusieurs fois une série d'instructions.

i est le **compteur** de la boucle ; il prend successivement les valeurs 1, 2, 3 … puis N.
La boucle se répète tant que le compteur *i* n'a pas atteint la valeur finale, ici N.
Une fois cette valeur prise par *i*, on sort de la boucle. On passe alors à la suite de l'algorithme.
Pour désigner la variable qui contrôle le nombre d'itérations (répétitions), on peut choisir une autre lettre que *i*.

Le nombre d'itérations est connu à l'avance ; on parle de **boucle « Pour »**.

Pour rendre la lecture plus claire, toutes les actions à effectuer au sein de la boucle sont **écrites en retrait** (on parle d'« indentations »).

La variable *i* peut ne pas commencer à 1 ; elle peut commencer à partir de n'importe quel nombre entier (positif ou négatif).

Le choix de la lettre *i* pour la variable de boucle vient de itération.

IV. Programmation en Python

Le mot « pour » se traduit « for » en anglais. En programmation, on parle de boucle « For ».

Le langage Python contient la fonction range.

L'instruction $\text{range}(a, b, \text{pas})$ donne la liste d'entiers relatifs de l'intervalle $[a\,;b[$ allant de pas en pas.
Par défaut (si rien n'est précisé) $a=0$ et $\text{pas}=1$, $L=[k \text{ for } k \text{ in range}(3)]$ donne la liste L des entiers *k* dans l'intervalle $[0\,;3[$, donc $L=[0,1,2]$.
Attention, en Python, les crochets ne signifient pas intervalle !

En langage naturel
Pour *i* variant de 1 à *n* **Faire**   instructions **FinPour**

En langage Python
`for i in range(1,n+1):`   [instructions]

Langage naturel
On utilise une barre d'indentation.

Langage Python
On doit faire à bien écrire les instructions du bloc for en décalage (indentation). Il faut penser aux deux points à la fin de la ligne.

Quand *i* n'intervient pas dans le bloc d'instructions, on peut remplacer l'instruction « *i* variant de 1 à *n* » par « *i* variant de 0 à $n-1$ » ou « *i* variant de 2 à $n+1$ ».
i doit parcourir un ensemble de n éléments.

Lorsque l'on écrit un programme en Python, il est indispensable de bien respecter les indentations. Si on oublie de décaler, on obtient un message d'erreur lorsqu'on le fait tourner.
Les indentation permettent une meilleure lisibilité du programme : on repère ainsi très bien le début et la fin de la boucle.

Il n'existe pas d'instruction pour définir la fin de la boucle. C'est l'indentation, c'est-à-dire le décalage vers la droite d'une ou plusieurs ligne, qui permet de marquer la fin de la boucle.

On peut écrire deux programmes : le premier correspond au **I**, le deuxième correspond au **II** sous forme d'une fonction ayant pour argument N, nombre de mois.

1er programme (pour 36 mois)

```
S=50
M=15
for i in range(1,37):
    S=S+M
    M=M+1
print (S)
```

2e programme (où l'on rentre le nombre N de mois)

```
def somme(N):
    S=50
    M=15
    for i in range(1,N+1):
        S=S+M
        M=M+1
    return S
```

3e programme (où l'on rentre le nombre N de mois)

```
def somme(N):
    S=50
    M=15
    print (S)
    for i in range(1,N+1):
        S=S+M
        M=M+1
        print (S)
```

La somme totale versée sur le livret de Léa au bout de 3 ans (c'est-à-dire le total de tous les versements effectués par les parents durant 3 ans) est de 1220 €.

La somme versée sur le livret de Léa au bout de 1 an sera de **296 €.**

La somme versée sur le livret de Léa au bout de 10 ans sera de **8 990 €.**

Exercice guidé

(faire fonctionner un algorithme avec boucle « Pour » à la main).

On considère l'algorithme suivant rédigé en langage naturel.

Variables :
n, i : entiers naturels avec $n \geqslant 1$
u : réel

Entrée :
Saisir n

Initialisation :
u prend la valeur 0

Traitement :
Pour i variant de 1 à n * **Faire**
    u prend la valeur $u + \frac{1}{i}$
FinPour

Sortie :
Afficher u

* avec un pas de 1

Recopier et compléter le raisonnement suivant :

Dans cet algorithme, i est … .

Lorsque l'utilisateur entre la valeur $n = 3$, i prend successivement les valeurs …, … , … .

La valeur initiale de u est … .

On a alors :

- pour $i = 1$, u prend la valeur $... = \ldots$

- pour $i = 2$, u prend la valeur $... = \ldots$

- pour $i = 3$, u prend la valeur $... = \ldots$

Conclusion : Lorsque l'utilisateur entre la valeur $n = 3$, l'algorithme affiche en sortie $u = ...$.

Solution

Cet exercice a pour objectif de comprendre le fonctionnement d'un algorithme simple avec une boucle « Pour » en l'effectuant « à la main ».

Dans cet algorithme, i est **la variable de boucle**.

Lorsque l'utilisateur entre la valeur $n = 3$, i prend successivement les valeurs **1**, **2**, **3**.

En effet, i peut prendre la valeur 1 (minimum) à la valeur 3 (maximum).

La valeur initiale de u est **0**.

On a alors :

- pour $i = 1$, u prend la valeur $0 + \frac{1}{1} = 1$.
- pour $i = 2$, u prend la valeur $1 + \frac{1}{2} = \frac{3}{2}$.
- pour $i = 3$, u prend la valeur $\frac{3}{2} + \frac{1}{3} = \frac{11}{6}$.

On évite d'employer le signe = (utilisé un peu abusivement pour i, mais pas utilisé pour u).
On utilise l'expression « prend la valeur ».

Ainsi la variable (le contenu de la variable) u évolue au fur et à mesure du déroulement de la boucle.

Conclusion : Lorsque l'utilisateur entre la valeur $n = 3$, l'algorithme affiche en sortie $\boxed{u = \frac{11}{6}}$ (dernière valeur de la variable u obtenue dans l'algorithme).

L'objectif est de savoir répondre directement à la dernière question.

On doit alors « dérouler » l'algorithme à la main.

Pour faire tourner « à la main » un algorithme avec une boucle « Pour », il est préconisé d'utiliser un tableau d'évolution des variables sur le modèle suivant :

i		1	2	3
u	0	1	$\frac{3}{2}$	$\frac{11}{6}$

On observera que la variable n n'apparaît pas dans le tableau car son contenu, fixé à 3 dès l'entrée, n'évolue pas au fur et à mesure du déroulement de l'algorithme. Il s'agit donc d'un tableau d'évolution des variables i et u.

Rédaction pour une question du type

« Faire tourner l'algorithme à la main pour $n = 3$ ».

Rédaction trouvée dans une copie du bac de juin 2012 :
« Déroulons l'algorithme pour $n = 3$. »

Le 27-8-2013

Comprendre le fonctionnement de l'algorithme (de la boucle) :
On répète … fois la même instruction.
La boucle contient … itérations.

Point-méthode :

Lorsque l'on fait fonctionner « à la main » une boucle « Pour », on peut utiliser un tableau.

Commentaires :

Comprendre qu'une boucle se déroule en plusieurs étapes.

Le contenu (c'est-à-dire les valeurs) des variables i et u évolue au fur et à mesure du déroulement de l'algorithme.

Il faut bien comprendre que l'on reprend toujours la valeur précédente de u c'est-à-dire que l'on remplace chaque fois u par sa valeur précédente.

Il est intéressant de dresser un tableau d'évolution des variables de l'algorithme permettant de voir le contenu des variables u et i au fur et à mesure du déroulement de l'algorithme.

On peut ainsi suivre le déroulement de l'algorithme « pas à pas ».

	Contenu de u
initialement	0
$i = 1$	1
$i = 2$	$\frac{3}{2}$
$i = 3$	$\frac{11}{6}$

```
def somme(n):
    u=0
    for i in range(1,n+1):
        u=u+1/i
    return u
```

Résumé du chapitre

1. Une boucle permet de **répéter** une instruction (ou une liste d'instructions) plusieurs fois.

La boucle pour répéter une instruction un certain nombre de fois (connu à l'avance) s'appelle **la boucle « Pour »**.

Le passage dans une boucle est appelé **itération**.

La **boucle « Pour »** se caractérise par le fait que l'on connaît à l'avance le nombre d'itérations que l'on va devoir effectuer.

2. Syntaxe

Pour (variable) allant de (début) à (fin) **Faire**

    Instructions

FinPour

Une boucle commence par un **Pour** et s'achève par un **FinPour**.

3. Elle fait intervenir une variable appelée **variable de boucle** ou **compteur**. C'est un entier naturel ; il faut définir son « minimum » et son « maximum ».

4. Dans un algorithme avec boucle, on n'arrive pas toujours à distinguer les étapes de l'algorithme.
On ne retrouve pas toujours la structure qui a été présentée dans le premier chapitre sur les algorithmes.
En particulier, la sortie peut s'effectuer au fur et à mesure des passages dans la boucle.

5. Il y a souvent une étape d'**initialisation** de certaines variables.

6. Les boucles « Pour » s'utilisent dans des algorithmes de calcul (**calculs itératifs**) mais pas seulement (algorithmes de simulation, algorithmes géométriques). Voir exercices.

Objectif du chapitre :

Comprendre le principe d'une boucle « Pour » : une seule instruction qui sera répétée un certain nombre de fois (donné à l'avance)

Savoirs

À la fin du chapitre, il faut avoir compris :

- l'intérêt et le fonctionnement d'une boucle « Pour » (gain de temps en automatisant le calcul)

Plus précisément :

- le rôle de la variable de boucle (compteur, entier naturel) qui intervient ou non dans les calculs dans la boucle

- la notion de début et de fin de boucle

- la notion d'itération

- l'évolution des variables après chaque « passage » dans la boucle (éventuellement en utilisant un tableau de suivi des variables)

- la procédure d'affichage qui peut s'effectuer au fur et à mesure du déroulement de la boucle (à chaque passage dans la boucle) ou bien à la fin de la boucle

- la notion d'initialisation d'une (ou de plusieurs) variable(s)

À la fin du chapitre, il faut connaître les mots-clefs :

- **itération** (nombre d'itérations)

- **variable de boucle** (**compteur**)

- **initialisation** (initialiser une ou plusieurs variables)

Une application intéressante : le paradoxe des anniversaires.

Savoirs-faire :

- comprendre un algorithme donné comprenant une boucle « Pour »

- modifier un algorithme avec une boucle « Pour »

- créer un algorithme avec une boucle « Pour »

- savoir réaliser le programme avec boucle « Pour » sur calculatrice et sur ordinateur

- connaître l'architecture générale (le squelette) d'un algorithme de calcul de somme et connaître le programme correspondant (éventuellement en ayant le squelette d'un tel programme dans sa calculatrice, cf. exercices)

- savoir interpréter l'utilité d'un algorithme avec boucle « Pour » par rapport à une situation « concrète » donnée

- savoir réécrire les algorithmes des exemples du cours

- savoir faire fonctionner à la main une boucle « Pour » pour un petit nombre d'itérations donné

- comprendre les étapes d'une boucle « Pour » (notamment comprendre l'évolution des variables dans une boucle « Pour »)

Exercices sur les boucles « Pour »

Les exercices s'articulent autour des compétences suivantes : comprendre, expliquer, interpréter, modifier, écrire, programmer un algorithme avec une boucle « Pour ».

Dans les exercices où un algorithme est donné, il est demandé de le recopier pour s'imprégner de la rédaction.

1 On considère l'algorithme suivant rédigé en langage naturel.

Variables :
n, *i* : entiers naturels

Traitement et sorties :
Pour *i* variant de 1 à 9 **Faire**
    n prend la valeur $7 \times i$
    Afficher *n*
FinPour

Recopier cet algorithme.
Quels affichages (c'est-à-dire quelles valeurs numériques) obtient-on avec cet algorithme ?
On demande de faire fonctionner l'algorithme « à la main ».
On pourra dresser un tableau d'état des variables dans la boucle.

Rentrer le programme correspondant sur calculatrice et vérifier les résultats.

2 On considère l'algorithme suivant rédigé en langage naturel.

Variables :
n, *i* : entiers naturels

Initialisation :
n prend la valeur 1

Traitement :
Pour *i* variant de 1 à 9 **Faire**
    n prend la valeur $2 \times n$
FinPour

Sortie :
Afficher *n*

Recopier cet algorithme.

Quel affichage obtient-on avec cet algorithme (c'est-à-dire quelle est la valeur de *n* à la fin de l'algorithme) ?

3 On considère l'algorithme ci-dessous rédigé en langage naturel.

Variables :
A, *n*, *i* : entiers naturels

Entrée :
Saisir *n*

Initialisation :
A prend la valeur 1

Traitement :
Pour *i* allant de 1 à *n* **Faire**
    A prend la valeur $A \times i$
FinPour

Sortie :
Afficher A

Recopier cet algorithme.

Quel est le nombre de sortie si le nombre d'entrée *n* est égal à 7 ?
On peut éventuellement utiliser une calculatrice pour effectuer les calculs.

Programmer cet algorithme sur calculatrice et vérifier le résultat précédent.

4 **Algorithme de somme**

On considère l'algorithme suivant :

Entrée :
Saisir N (entier naturel)

Initialisation :
S prend la valeur 0

Traitement :
Pour K allant de 0 à N **Faire**
    S prend la valeur $S + K$
FinPour

Sortie :
Afficher S

Recopier cet algorithme.

1°) Quelle est la valeur affichée par S si on fait fonctionner « à la main » l'algorithme pour $N = 10$?

2°) Écrire le programme de calculatrice (en langage de la calculatrice) correspondant à l'algorithme.
Vérifier alors le résultat de la question précédente.

5 **Algorithme de somme**

1°) a) Écrire un algorithme en langage naturel qui permet de calculer la somme des carrés de tous les entiers naturels de 0 à 100 (c'est-à-dire $S = 0^2 + 1^2 + 2^2 + 3^2 + ... + 100^2$).

b) Programmer cet algorithme sur calculatrice (écrire le programme dans le langage de la calculatrice).

c) Donner la valeur de S obtenue.

2°) Écrire un algorithme qui demande un nombre entier naturel *n* à l'utilisateur et qui renvoie la somme des carrés de tous les entiers naturels de 0 à *n*.

6 On considère l'algorithme ci-dessous rédigé en langage naturel.
Celui-ci fait intervenir une liste L de taille *n*.

Entrée :
Saisir *n*

Traitement :
Boucle 1 :
Pour *i* variant de 1 à *n* **Faire**
    L[*i*] prend la valeur d'un entier aléatoire de 0 à 50 (au sens large)
FinPour
X prend la valeur 0
Boucle 2 :
Pour *i* variant de 1 à *n* **Faire**
    Si $L[i] > X$
        X prend la valeur L[*i*]
    FinSi
FinPour

Sortie :
Afficher X

Recopier cet algorithme.

1°) On se place dans le cas où $n = 5$.
On suppose que $L[1] = 0$, $L[2] = 15$, $L[3] = 10$, $L[4] = 30$, $L[5] = 20$.

a) Quelle est la valeur de X à la fin du premier passage dans la boucle 2 ?
b) Indiquer la valeur de X à la fin de chaque passage dans la boucle 2.
c) Quelle est la valeur de X affichée à la fin de l'algorithme ?

2°) Dans le cas général, quelle relation y a-t-il entre la valeur de X affichée à la fin de l'algorithme et les valeurs L[1], L[2], … L[*n*] ?

7 Un algorithme important : algorithme de tracé d'une courbe

But : comprendre ce que fait la calculatrice ou un logiciel de tracé de courbes

f est une fonction définie sur un intervalle $[a\,;b]$.

On étudie l'algorithme ci-dessous :

Entrées :
Saisir a, b : bornes de l'intervalle de définition
Saisir f : fonction étudiée
Saisir N : entier naturel ($N \geqslant 1$)

Initialisations :
pas prend la valeur $\dfrac{b-a}{N}$
x prend la valeur a

Traitement et sorties :
Pour k de 0 jusqu'à N **Faire**
  Marquer le point de coordonnées $(x\,;\,f(x))$
  x prend la valeur $x+$ pas
FinPour

On notera que le nom de la variable « pas » (on pourrait tout aussi bien la désigner par une lettre par exemple p).

Faire fonctionner cet algorithme « à la main » dans le cas où f est la fonction définie sur $[0\,;1]$ par $f(x)=x^2$ et $N=10$.
Présenter les résultats dans un tableau du type :

x	0	...	
$f(x)$	...	...	

Expliquer le fonctionnement et l'utilité de cet algorithme.

8 1°) On considère l'algorithme suivant :

Traitement et sortie :
Pour X allant de – 5 à 5 avec un pas de 1* **Faire**
  A prend la valeur X^2
  B prend la valeur $2X+8$
  Si $A \leqslant B$
    Alors afficher X
  FinSi
FinPour

* « avec un pas de 1 » signifie de 1 en 1.

Recopier cet algorithme.

On peut faire tourner l'algorithme « à la main ».

a) Que fait cet algorithme ?
b) Programmer cet algorithme sur calculatrice.
2°) Afficher sur l'écran de la calculatrice la courbe représentative de la fonction « carré » ainsi que la droite d'équation $y=2x+8$. Retrouver sur le graphique le résultat précédent.
3°) Retrouver le résultat précédent par le calcul.

9 Faisceau de droites

1°) Entrer le programme ci-dessous dans la calculatrice (on prendra soin d'effacer les fonctions déjà présentes).

Programme sur calculatrice TI

```
: EffDessin
: For (A,– 3,5,0.5)
: DessFonct (AX – A)
: End
```

Ou

```
: ClrDraw
: For (A,– 3,5,0.5)
: DrawF(AX – A)
: End
```

Programme sur calculatrice CASIO

```
ClrGraph↵
For – 3 → A To 5 Step 0.5 ↵
Graph Y = AX – A ↵
Next↵
```

Indications pour calculatrice TI 83+ :

On utilise les commandes de dessin obtenues en faisant : 2nde prgm .

Indications pour calculatrice Casio Graph 35 + :

- Pour effacer les fonctions précédentes : Clr Graph : SHIFT VARS F6 F1 F2
- Pour afficher la fonction : Graph Y = : SHIFT F4 F5 F1

2°) Exécuter ce programme.

3°) Faire une conjecture d'après le graphique obtenu à la question 2°).
4°) Démontrer la conjecture.

10 L'algorithme ci-dessous se réfère aux lancers d'un dé cubique non truqué dont les faces sont numérotées de 1 à 6.

Entrée :
Saisir n (nombre de lancers)

Initialisation :
s prend la valeur 0

Traitement :
Pour i variant de 1 à n **Faire**
  k prend la valeur d'un entier aléatoire entre 1 et 6 au sens large
  Si $k = 6$
    Alors s prend la valeur $s+1$
  FinSi
FinPour

Sortie :
Afficher s

Recopier cet algorithme.
À quoi correspond la valeur de s obtenue en sortie ?
Programmer cet algorithme.

11 On étudie les fréquences d'apparition du côté Pile d'une pièce de monnaie parfaitement équilibrée lors de lancers successifs.
On lance N fois la pièce : on obtient un échantillon de taille N, puis on répète E fois cette expérience.
1°) Écrire un algorithme permettant d'obtenir E échantillons de N lancers avec la fréquence.
Utiliser deux listes L_1 et L_2 : la liste L_1 contiendra le numéro de l'expérience et la liste L_2 contiendra les fréquences de Pile.
2°) Programmer cet algorithme sur calculatrice.
Exécuter cet algorithme pour $N = 20$ et $E = 50$, $N = 100$ et $E = 50$, $N = 200$ et $E = 50$.
Représenter le nuage de points associé.

12 On considère l'algorithme suivant dans le plan muni d'un repère.

Variables :
a, b, i, y

Entrées :
Saisir a et b

Traitement et sorties :
Pour i variant de 1 à 10 **Faire**
  y reçoit la valeur $a \times i + b$
  Marquer le point de coordonnées $(i\,;\,y)$
FinPour

Recopier cet algorithme.

1°) Préciser ce que l'on obtient en exécutant cet algorithme.
2°) Le fait d'inverser l'ordre des deux dernières instructions modifie-t-il le résultat ?
3°) Programmer cet algorithme sur calculatrice et le faire fonctionner pour différentes valeurs de a et b.

13 Une expérience consiste à lancer 10 fois de suite une pièce équilibrée et à dénombrer les sorties Pile ou Face. On se propose de simuler n fois cette expérience (de 10 lancers). Pile est associé à 0 et Face à 1.

Variables :
c, N, i, k, r, S

Entrée :
Afficher « nombre d'expériences ? »
Saisir N

Initialisation :
c prend la valeur 0

Traitement :
Pour i variant de 1 à N **Faire**
  S prend la valeur 0
  Pour k variant de 1 à 10 **Faire**
    r prend la valeur 0 ou 1 au hasard
    S prend la valeur $S + r$
  FinPour
  Si $S = 5$
    Alors c prend $c+1$
  FinSi
FinPour

Sortie :
Afficher $\dfrac{c}{N}$

Recopier cet algorithme.

1°) Analyser l'algorithme et préciser ce qu'il affiche en sortie.

2°) Répondre à la même question après avoir remplacé « Si $S = 5$ » par « Si $S \leqslant 3$ »

14 1°) Rédiger un algorithme qui demande à l'utilisateur une valeur de n, entier naturel supérieur ou égal à 1, et qui affiche en sortie la valeur de $\sum_{p=1}^{n}\sum_{q=1}^{n}|p-q|$.

2°) Démontrer que $\sum_{p=1}^{n}\sum_{q=1}^{n}|p-q| = \dfrac{n\left(n^2-1\right)}{3}$.

15 On considère les suites (a_n) et (b_n) définies
- par leurs premiers termes $a_0 = 5$ et $b_0 = 7$;
- par les relations $a_{n+1} = a_n + b_n$ et $b_{n+1} = 2a_n b_n$.

Écrire un algorithme qui demande à l'utilisateur un entier naturel $n \geqslant 1$ puis qui calcule et affiche les valeurs de a_n et b_n.

16 Écrire un algorithme qui demande à l'utilisateur un entier naturel $n \geqslant 2$ puis qui calcule et affiche la valeur de $\sum_{1 \leqslant i \leqslant j \leqslant n} \frac{1}{i+j}$.

17 On s'intéresse au nombre de carrés parfaits inférieurs ou égaux à un entier naturel *n*.

1°) a) Compléter l'algorithme ci-dessous qui permet à un utilisateur de saisir en entrée un entier naturel *n* et qui affiche en sortie le nombre de carrés parfaits inférieurs ou égaux à *n*.

Entrée :
Saisir *n*

Initialisation :
c prend la valeur 0

Traitement :
Pour *k* allant de 0 à *n* **Faire**
  Si $\sqrt{k}$ est un entier
    Alors *c* prend la valeur ……
  FinSi
FinPour

Sortie :
Afficher *c*

b) Réaliser le programme correspondant sur calculatrice et donner le résultat obtenu en sortie si l'on saisit 2012 en entrée.
2°) Donner une formule directe donnant le nombre de carrés parfaits inférieurs ou égaux à un entier nature n en utilisant la partie entière.

18 Écrire un algorithme qui demande deux entiers naturels *n* et *m* (*n* inférieur ou égal *m*) et qui calcule et affiche la somme de tous les entiers naturels compris entre *n* et *m* au sens large.

19 On donne l'algorithme de calcul suivant :

Entrée :
Saisir *n*

Initialisation :
u prend la valeur 1

Traitement :
Pour *k* allant de 0 à *n* **Faire**
  u prend la valeur $u + 10^k$
FinPour

Sortie :
Afficher *u*

1°) Faire fonctionner cet algorithme pour $n = 8$.
Quel est le nombre affiché ? Justifier.
2°) Modifier l'algorithme pour que le nombre affiché soit 25252525…25 avec *n* tranches de « 25 ».

Corrigé

1

Variables :
n, i : entiers naturels

Traitement et sorties :
Pour i variant de 1 à 9 **Faire**
    n prend la valeur $7 \times i$
    Afficher n
FinPour

Il s'agit d'un algorithme de calcul. C'est un algorithme sans utilité qui permet juste de comprendre comment fonctionne un algorithme avec une boucle Pour.

L'énoncé demande de faire fonctionner l'algorithme à la main dans un premier temps et de le programmer dans un deuxième temps (il est évidemment plus rapide de le programmer tout de suite, mais l'intérêt de l'exercice est de comprendre le fonctionnement d'une boucle « Pour »).

Cet algorithme comprend une boucle « Pour » avec sortie au fur et à mesure du déroulement de l'algorithme.

Dans cet algorithme, la variable de boucle i intervient dans les calculs.

On a choisi 9.
Il faut donc faire varier i de 1 à 9.
Il faut juste écrire les valeurs prises par n (valeurs qui vont s'afficher au fur et à mesure du déroulement de l'algorithme).

$i = 1$ n prend la valeur $7 \times 1 = 7$ (on n'écrit pas $n = 7 \times 1 = 7$)
$i = 2$ n prend la valeur $7 \times 2 = 14$
$i = 3$ n prend la valeur $7 \times 3 = 21$
$i = 4$ n prend la valeur $7 \times 4 = 28$
$i = 5$ n prend la valeur $7 \times 5 = 35$
$i = 6$ n prend la valeur $7 \times 6 = 42$
$i = 7$ n prend la valeur $7 \times 7 = 49$
$i = 8$ n prend la valeur $7 \times 8 = 56$
$i = 9$ n prend la valeur $7 \times 9 = 63$

Cet algorithme affiche la **table de multiplication par 7** (ça fait revoir la table de 7 !).
On obtient tous les multiples de 7 jusqu'à 9×7.

On peut présenter les résultats dans un tableau.

i	1	2	3	4	5	6	7	8	9
n	7	14	21	28	35	42	49	56	63

On peut aussi présenter ce tableau verticalement.

La variable de boucle i intervient dans les calculs.

Programmation sur calculatrice :

Programme sur calculatrice TI

```
: For (I,1,9)
: 7 × I → N
: Disp N
: End
```

Programme sur calculatrice CASIO

```
For 1 → I To 9↵
7×I →N↵
N◢
Next↵
```

Appuyer sur la touche EXE à chaque fois pour voir apparaître toute la table de 7.

On remarquera que l'écriture des programmes respecte les lettres de l'algorithme à la seule différence que les lettres minuscules sont transformées en lettres majuscules.

On peut éventuellement ajouter Pause avant le End de manière à ce que l'on ait le temps de voir tous les résultats.

2 Il s'agit d'un algorithme de calcul.

Dans cet algorithme, on repère tout de suite la variable de boucle : i (compteur de la boucle).

Variables :
n, i : entiers naturels

Initialisation :
n prend la valeur 1

Traitement :
    Pour i variant de 1 à 9 **Faire**
        n prend la valeur $2 \times n$
    FinPour

Sortie :
Afficher n

Dans cet algorithme :

- la variable de boucle *i* n'intervient pas dans les calculs ;

- l'affichage s'effectue à la fin du déroulement de l'algorithme.

Dans l'initialisation, *n* prend la valeur 1 : c'est la valeur de départ de *n*.

L'algorithme se déroule en 9 étapes (algorithmes avec 9 étapes) : on répète 9 fois la même instruction.
On commence par $i = 1$, on calcule la valeur de *n*.
On passe ensuite à $i = 2$, on calcule ensuite la valeur de *n*…
Et ainsi de suite jusqu'à $i = 9$.

La valeur de la variable *n* va évoluer au fur et à mesure du passage dans les boucles.
$i = 1$ ($1^{ère}$ boucle) : *n* prend la valeur $2 \times 1 = 2$
$i = 2$ (2^{e} boucle) : *n* prend la valeur $2 \times 2 = 4$ (on reprend la valeur de *n* à la fin de la boucle précédente)
$i = 3$ (3^{e} boucle) : *n* prend la valeur $2 \times 2 \times 2$
etc.

$i = 9$ (9^{e} boucle) : *n* prend la valeur $2 \times 2 \times \ldots \times 2$ (9 facteurs)

L'algorithme affiche en sortie $2^9 = 512$ (valeur de la variable *n* en sortie).

Programmation sur calculatrice :

Programme sur calculatrice TI

```
: 1 → N
: For (I,1,9)
: 2 × N → N
: End
: Disp N
```

Programme sur calculatrice CASIO

```
1 → N↵
For 1 →I To 9↵
2 × N →N ↵
Next↵
N◢
```

3 Il s'agit d'un **algorithme de calcul**.

Dans cet algorithme :

- la variable de boucle *i* intervient dans les calculs ;

- l'affichage est effectué à la fin du déroulement de l'algorithme.

On cherche le nombre de sortie si le nombre d'entrée *n* est égal à 7.

La valeur de A évolue au fur et à mesure de la progression de l'algorithme. Il change à chaque passage dans la boucle.

La valeur prise initialement par A est 1.

***i* = 1.** Valeur de A après le 1^{er} passage dans la boucle : $1 \times 1 = 1$

***i* = 2.** Valeur de A après le 2^{e} passage dans la boucle : $1 \times 2 = 2$

***i* = 3.** Valeur de A après le 3^{e} passage dans la boucle : $2 \times 3 = 6$

***i* = 4.** Valeur de A après le 4^{e} passage dans la boucle : $6 \times 4 = 24$

***i* = 5.** Valeur de A après le 5^{e} passage dans la boucle : $24 \times 5 = 120$

***i* = 6.** Valeur de A après le 6^{e} passage dans la boucle : $120 \times 6 = 720$

***i* = 7.** Valeur de A après le 7^{e} passage dans la boucle : $720 \times 7 = 5040$

Conclusion : Si le nombre d'entrée est 7, le nombre de sortie est 5040.

Programme sur calculatrice TI

```
: Prompt N
: 1 → A
: For (I,1,N)
: A × I → A
: End
: A
```

Programme sur calculatrice CASIO

```
"N="?→N↵
1 → A ↵
For 1→I To N↵
A×I→A ↵
Next↵
A◢
```

Commentaire :

La valeur de A affichée à la fin de l'algorithme est $1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7$.
Ce produit se note 7 ! (7 suivi d'un point d'exclamation).
On lit « factorielle de 7 » ou « 7 factorielle ».
Les anglais lisent, eux, « 7 admiration ».
Il y avait eu une autre notation en usage au $XVIII^{e}$ siècle jusqu'à ce qu'un professeur de Metz du début du XIX^{e} siècle invente la notation avec le point d'exclamation. C'est finalement cette dernière notation qui a été retenue.

Sur la calculatrice, on peut obtenir la factorielle de n'importe quel nombre entier.

Sur calculatrice TI, taper le nombre 7 puis, afin d'obtenir le point d'exclamation, aller dans MATH, PRB et sélectionner la ligne 4 (avec le point d'exclamation).

Sur calculatrice Casio Graph 35+, taper le nombre 7 puis, afin d'obtenir le point d'exclamation, appuyer sur la touche OPTN puis sélectionner **PROB** (probabilités) puis sélectionner **x !**.

Programme sur calculatrice TI

```
: Prompt N
: 0 → S
: For(K,0,N)
: S + K → S
: End
: Disp S
```

Programme sur calculatrice CASIO

```
"N="?→ N↵
0 → S↵
For 0 → K To N↵
S + K → S↵
Next↵
S ◢
```

Attention à la place du « End » et du « Disp S ». On peut les intervertir à la fin du programme.

On pourra remarquer qu'une telle somme peut être obtenue sur calculatrice en utilisant les fonctionnalités de la calculatrice qui permettent de calculer des sommes de suites.

4 Il s'agit d'un **algorithme de calcul de somme**.

1°) On effectue les calculs suivants :

$0+0=0$
$0+1=1$
$1+2=3$
$3+3=6$
$6+4=10$
$10+5=15$
$15+6=21$
$21+7=28$
$28+8=36$
$36+9=45$
$45+10=55$

Conclusion : Si le nombre d'entrée est 10, le nombre de sortie est 55.

L'algorithme permet de calculer $\sum_{k=0}^{k=10} k = 0+1+2+3+4+5+6+7+8+9+10$.

1°) Pour N = 10

$S=\left(\left(\left(\left(\left(\left(\left(\left(\left(\left(0+0\right)+1\right)+2\right)+3\right)+4\right)+5\right)+6\right)+7\right)+8\right)+9\right)+10\right)$

S = 55

2°) Avec le programme sur calculatrice, pour N = 10, on retrouve S = 55.

2°) **Programmes sur calculatrice**

Programme sur calculatrice TI

```
: Prompt N
: 0 → S
: For (K,0,N)
: S + K → S
: End
: Disp S
```

Programme sur calculatrice CASIO

```
"N="? →N↵
0 → S ↵
For 0 → I To N↵
S + K → S ↵
Next↵
S◢
```

5 Algorithme de somme

1°) a) Algorithme rédigé en langage naturel qui permet de calculer la somme des carrés de tous les entiers naturels de 0 à 100 (c'est-à-dire $S=0^2+1^2+2^2+3^2+...+100^2$).

Initialisation :
S prend la valeur 0

Traitement :
Pour K allant de 0 à 100 **Faire**
  Affecter à S la valeur $S+K^2$
FinPour

Sortie :
Afficher S

Explication :

On part de 0.
On fait $0+0^2=0$.
On fait $0+1^2=1$.
On fait $1+2^2=5$.
On fait $5+3^2=14$ etc.

b) **Programme sur calculatrice**

Programme sur calculatrice TI

```
: 0 → S
: For(K,0,100)
: S + K² → S
: End
: Disp S
```

Programme sur calculatrice CASIO

```
0 → S↵
For 0 → K To
100↵
S + K² → S↵
Next↵
S ◢
```

c) La valeur de S obtenue en sortie est **338 350**.

2°) Algorithme qui demande un nombre entier naturel *n* à l'utilisateur et qui renvoie la somme des carrés de tous les entiers naturels de 0 à *n*.

Entrée :
Saisir *n*

Initialisation :
S prend la valeur 0

Traitement :
Pour K allant de 0 à *n* **Faire**
  Affecter à S la valeur $S+K^2$
FinPour

Sortie :
Afficher S

Finalement, autant faire les deux en même temps.

Programme sur calculatrice TI

```
: Prompt N
: 0 → S
: For(K,0,N)
: S+K² → S
: End
: Disp S
```

Programme sur calculatrice CASIO

```
"N="?→ N↵
0 → S↵
For 0 → K To N↵
S+K² → S↵
Next↵
S ◢
```

6 Algorithme avec une liste

Entrée :
Saisir *n*

Traitement :
Boucle 1
Pour *i* variant de 1 à *n* **Faire**
  L[*i*] prend la valeur d'un entier aléatoire de 0 à 50 (au sens large)
FinPour
X prend la valeur 0
Boucle 2
Pour *i* variant de 1 à *n* **Faire**
  Si L[*i*] > X
    X prend la valeur L[*i*]
  FinSi
FinPour

Sortie :
Afficher X

L'intérêt de cet algorithme est de travailler sur les listes (création d'une liste avec un algorithme).

Analyse de l'algorithme :

- Les deux boucles sont indépendantes l'une de l'autre.
- La boucle 1 sert à rentrer une liste de *n* nombres entiers aléatoires compris entre 0 et 50 au sens large.

$1 \leqslant i \leqslant n \qquad 0 \leqslant L[i] \leqslant 50$

- Le but de l'exercice est de déterminer le rôle de boucle 2 dans l'algorithme (cf. suite des questions de l'exercice).
- Du point de vue de l'écriture de l'algorithme, on pourrait utiliser une autre variable que *i* (utilisée dans la boucle 1) pour la boucle 2. C'est ce qui sera fait dans le programme pour calculatrice Casio.

1°) On se place dans le cas où $n = 5$.

On suppose que à la fin de la boucle 1, la liste est $L = [0,15,10,30,20]$.

$L[1] = 0$, $L[2] = 15$, $L[3] = 10$, $L[4] = 30$, $L[5] = 20$

a) **Valeur de X à la fin du premier passage dans la boucle 2**

Au départ, X a été initialisé à 0.

$i = 1$
$L[1] = 0$
L'inégalité $L[1] > X$ est fausse (puisque L[1] et X ont tous les deux la valeur 0).
Donc, après le premier passage dans la boucle, la valeur de X n'a pas changé ; elle est toujours égale à 0.

b) **Valeur de X à la fin de chaque passage dans la boucle 2**

Il faut bien comprendre qu'à chaque fois, on doit prendre en compte la valeur précédente.

$i = 2$
$L[2] = 15$
L'inégalité $L[2] > X$ est vraie.
Donc après le deuxième passage dans la boucle, la valeur de X est 15.

$i = 3$
$L[3] = 10$
L'inégalité $L[3] > X$ est fausse.
Donc après le troisième passage dans la boucle, la valeur de X est 15.

$i = 4$
$L[4] = 30$
L'inégalité $L[4] > X$ est vraie.
Donc après le quatrième passage dans la boucle, la valeur de X est 30.

$i = 5$
$L[5] = 20$
L'inégalité $L[5] > X$ est fausse.
Donc après le cinquième passage dans la boucle, la valeur de X est 30.
On peut aussi présenter l'évolution des variables *i* et X dans un tableau.

i	X
$i = 1$	X = 0 0
$i = 2$	X = 15
$i = 3$	X = 15
$i = 4$	X = 30
$i = 5$	X = 30

c) **Valeur de X affichée à la fin de l'algorithme**

La valeur de X affichée à la fin de l'algorithme est 30.

2°) **Relation la valeur de X affichée à la fin de l'algorithme et les valeurs L[1], L[2], … L[*n*]**

La valeur de X affichée à la fin de l'algorithme est la valeur maximale de la liste aléatoire de nombres entiers compris entre 0 et 50 au sens large.

Ou :

X est la plus grande valeur de la liste.

Le but de l'algorithme est d'afficher la plus grande valeur de la liste.

Partie programmation :

Il est plus prudent d'effacer toutes les listes avant d'éditer le programme.

• Programme pour calculatrice TI

Sur calculatrice, pour obtenir un entier aléatoire compris entre 0 et 50 au sens large, on peut taper :
partEnt(51 * NbrAléat) ou entAléat(0,50).

On doit utiliser la partie entière notée partEnt obtenue MATH NUM.
Pour obtenir NbrAléat, aller dans MATH PRB.

S'assurer au départ que la liste L_1 est vide.


```
: EffListe L1
: Prompt N
: For (I,1,N)
: partEnt(51 * NbrAléat) → L1(I)
: End
: 0 → X
: For (I,1,N)
: If L1(I) > X
: Then
: L1(I) L1(I) → X
: End
: Disp X
```

On obtient L1 en tapant [2nde] ou [2nde] [stats] (listes).

Pour effacer une liste, [STAT] et 4

4 : ClrList (anglais) ou 4 : EffListe (français)

J'avais pensé utiliser une séquence du type suite (partEnt(51*NbrAléat),K,1,N).

• Programme pour calculatrice CASIO :

On obtient la partie entière (notée Int) de la manière suivante : [OPTN] puis [NUM] (touche [F5]) puis [Int] (touche [F5]).

```
"N="?→ N↵
For 1 → I To N↵
Int(51×Ran#) → List 1[I]↵
Next
For 1 → J To N↵
If List 1[J] > X
Then List 1[J] → X
IfEnd
Next
X◢
```

Utilisation du maximum d'une liste :

Sur calculatrice, on peut obtenir le maximum d'une liste.
Du coup, le programme se simplifie considérablement puisque le travail de la boucle 2 est remplacé par une seule instruction.

Calculatrice TI :

```
: Prompt N
: For (I,1,N)
: partEnt(51 * NbrAléat) → L1(I)
: End
: max(L1)→ X
: Disp X
```

Pour effacer une liste :

Calculatrice CASIO :

La touche max s'obtient en appuyant sur la touche OPTN puis en sélectionnant LIST puis max (tout en bas de l'écran).

Certains langages de programmation ont une commande qui permet de trouver le maximum d'une liste.

On peut aussi rédiger un algorithme sans liste :

```
: Prompt N
: 0 → X
: For(I,1,N)
: PartEnt(51* NbrAléat) → Y
: If Y > X
: Y → X
: End
: End
: Disp X
```

À quoi sert cet algorithme ?

Cet algorithme n'a pas grande utilité en soi. Il a surtout pour but de montrer l'utilisation de listes.

Cependant, il permet de traiter des simulations dans d'autres cas (exemple : on lance trois dés cubiques non triqués, on note le plus grand des numéros des faces supérieures).

7 Algorithme de tracé d'une courbe

Il s'agit d'un algorithme géométrique.
Il est intéressant de comprendre ce que fait la calculatrice (ou un logiciel de tracé de courbe) : elle trace la courbe à l'aide de segments (même si on le sait déjà, du moins on s'en doute).

$f(x) = x^2$

$a = 0$; $b = 1$

$N = 10$

$\text{pas} = \dfrac{b-a}{N} = 0{,}1$

	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
x	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$f(x)$	0	0,01	0,04	0,09	0,16	0,25	0,36	0,49	0,64	0,81	1

L'algorithme produit un nuage de points, pas une courbe.

Cet algorithme n'a vraiment d'intérêt que pour le tracé de la courbe représentative d'une fonction continue sur un intervalle (le sens du mot « continue » est précisé en Terminale ; en gros, la courbe doit pouvoir être tracée sans lever le crayon sur cet intervalle).
C'est le cas de la fonction « carré » qui est bien continue sur $\mathbb{R}$ et donc par restriction sur l'intervalle [0 ; 1].

Lorsque l'on demande à la calculatrice de tracer la courbe d'une fonction discontinue, il arrive fréquemment que la calculatrice relie à tort certains points comme s'il s'agissait de la courbe d'une fonction continue (« petits murs » que l'on peut faire disparaître grâce à une commande spéciale de la calculatrice).

Cet algorithme marche bien car la fonction « carré » est continue (au sens où l'on peut tracer sa courbe représentative sans lever le crayon).
Il ne marcherait pas ou mal pour une fonction non continue.

L'algorithme fonctionne car la fonction « carré » est une fonction continue.
Il marcherait mal si elle ne l'était pas (non traçable en un seul morceau).

Cet exercice est aussi l'occasion de signaler un point qui l'est rarement dans tous les cours : la courbe de la fonction « carré » n'est constituée d'aucun segment de droites. Ce résultat intuitif peut se démontrer.
En effet, si M et N sont deux points de la courbe de la fonction « carré », on peut démontrer que l'arc de courbe]MN[est strictement au-dessus du segment]MN[.

Sur calculatrice TI, on peut définir la fonction f dans [$f(x)$] (comme pour afficher le graphique) : Y_1 = … .

On introduit ensuite dans le programme dans [Var] , VAR–Y =, 1 : Fonction, Y_1.

On peut ensuite changer la fonction dans [$f(x)$] .

⚠ Dans le programme, il faut écrire manuellement (x) car il n'affiche que Y_1. On écrit donc $Y_1(x)$.

Programme sur calculatrice TI :

- Obtention du nuage de points :

```
: ClrDraw
: Prompt A
: Prompt B
: Prompt N
: (B – A) / N → P
: A→X
: For(K, 0, N)
: Point-On(X, X²)
: X + P→X
: End
```

Sur calculatrice Casio PlotOn X, Y

(PlotOn s'obtient grâce à CATALOG [SHIFT] [4])

On met une instruction au début du programme pour effacer les dessins précédents.

Essayer l'algorithme pour différentes valeurs de N.
L'algorithme permet de placer N + 1 points.

Plus N est grand, plus il y a de points, plus le tracé est précis.

Pour aller plus loin : tracé pour une fonction quelconque

- Obtention de la ligne brisée : à revoir il faut une instruction de tracé des segments

```
: ClrDraw
: Input " Y1 = ", Y1   (utiliser [VAR], [→], [Y-VARS], [1], [1])
: Prompt A, B, N
: If N ⩾ 1
: (B – A) / N → P
: A → X
: For(K, 0, N – 1)
: FonctNAff   (accessible à partir du catalogue, ou : [VAR], [→], [Y-Var], [4], [2])
: Ligne(X, Y1(X), X + P, Y1(X + P))
: X + P→X
: End
```


Le 3-1-2013

Pour Y1 dans le programme. On utilise la calculatrice (ce n'est pas à nous de taper le Y et le 1).

[var] Y-VARS 1 : Function 1 : Y1

Instruction « FonctNAff (accessible à partir du catalogue, ou : [VAR], [→], [Y-Var], [4], [2]) » à mettre en plein dans la boucle.

Sur TI :

Pour tracer des segments sur calculatrice, chercher dans catalogue **Ligne(** (pour les calculatrices en français) ou **Lign(** (pour les calculatrices en anglais).

Exemple : Ligne (2 ; 3 ; 0 ; 4) permet de tracer le segment joignant les points de coordonnées (2 ; 3) et (0 ; 4).
Sinon on peut utiliser les listes.

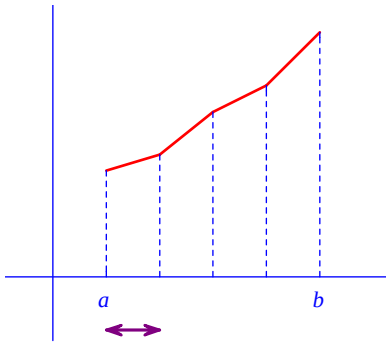
Calculatrice TI : rentrer une fonction

Input " Y1 = ", Y1

Pour taper Y1 dans le programme : VARS 1-VARS
1 : fonction
1 : Y1

Exécution du programme :
Y1 " …. " mettre les guillemets (inutile de mettre le signe =)

Cet algorithme permet de comprendre ce que fait une calculatrice lorsqu'on lui demande de tracer une courbe.
Il permet aussi de calculer la longueur d'une courbe.



$\frac{b-a}{N}$

On subdivise l'intervalle [a ; b] en N petits segments qui ont tous la même longueur : $\boxed{\frac{b-a}{N}}$.

L'idée est de remplacer la courbe par des petits segments (des cordes).

Les mathématiciens se sont penchés sur ce type de problème très tôt : par exemple pour calculer la longueur de courbe (la méthode consistant à utiliser une ficelle n'étant pas ce que l'on peut trouver de mieux).
Pour effecteur de tels calculs, ils ont développé des outils.
Ils ont aussi utilisé des méthodes de calcul approché : remplacer les courbe par des segments ou par des morceaux de tangente.

- Tracé d'une courbe point par point.
- Définition d'une courbe comme ensemble de points.
- Courbe = obtenue en faisant tendre N vers + ∞.

En fait, le tracé d'une courbe sur un écran ne s'effectue pas de cette manière-là. Le tracé s'effectue avec des pixels au moyen d'algorithmes (algorithme du point médian par exemple).

Le 21 mai 2013

J'avais noté sur une feuille :

algorithme tracé d'une courbe pas tout à fait juste.
la calculatrice utilise des pixels.