# Cours de 1ère S/ Algorithmique

Eric Dostal

aout 2013

# Table des matières

## 1.1 Avant la programmation

### Qu'est ce qu'un algorithme ?

**Définition 1** *Un algorithme est une succession d'instructions (aussi appelées commandes) et permettant la résolution d'un problème donné.*

**Remarque :**
Le terme d'algorithme vient du nom du mathématicien arabe du IX$^{e}$ siècle *Al Khuwarizmi* qui écrivit la première méthode systématique de résolution de certaines équations.

**Exemple :**

```
pour A allant de 1 à 10 par pas de 1
Stocker A^2 dans B
Afficher B
```

L'algorithme précédent calcule et affiche le carré des nombres de 1 à 10. Dans cet algorithme,`Stocker` $A^2$ `dans B` est une instruction.

### Qu'est ce qu'un langage de programmation ?

**Définition 2** *Un langage de programmation est un ensemble d'instructions et de règles syntaxiques compréhensible par l'ordinateur et permettant de créer des algorithmes. Un programme est la traduction d'un algorithme dans le langage de programmation utilisé.*

**Exemples :**
BASIC, PASCAL, C++, assembleur sont des langages de programmation pour ordinateurs. Dans ce cours nous utiliserons les langages de programmation associés aux calculatrices programmables Casio et Texas Instrument ainsi que le langage de programmation du logiciel libre et gratuit XCas téléchargeable à l'adresse *http* : $//www-fourier.ujf-grenoble.fr/\ parisse/giac_fr.html$ et le langage de programmation du logiciel libre et gratuit ALGOBOX téléchargeable à l'adresse suivante *http* : $//www.xm1math.net/algobox/down$

## Avant de programmer

### Créer ou modifier ou exécuter un programme

**Casio :**
Touche MENU puis choisir PRGM et :

- EDIT pour modifier un programme existant ;
- NEW pour créer un nouveau programme ;
- EXEC pour exécuter un programme.

**TI :**
Touche PRGM puis :

- EDIT pour modifier un programme existant ;
- NEW pour créer un nouveau programme ;
- EXEC pour exécuter un programme existant.

**Remarque :**
Après création d'un nouveau programme sur TI ou CASIO, entrer le nom du programme ; n'utiliser que les lettres (touches ALPHA + Lettre)

**Algobox :**
AlgoBox est un logiciel libre, multi-plateforme et gratuit d'aide à l'élaboration et à l'exécution d'algorithmes dans l'esprit des nouveaux programmes de mathématiques du lycée.

**XCas :**
L'édition d'un programme se fait dans la ligne de commande. Avant de commencer, aller dans le menu Cfg configuration du CAS et vérifier que l'onglet PROG STYLE est en mode XCAS. On pourra aussi aller dans Cfg Polices (Toutes) et choisir une police de taille 14 plus lisible que la police de taille 18 par défaut.

### Instructions d'un programme

**Casio :**
Les instructions des algorithmes peuvent être séparées par un retour à la ligne EXE. Une ligne peut éventuellement comporter plusieurs instructions séparées par :.

**TI :**
Les instructions des algorithmes peuvent être séparées par un retour à la ligne EXE. Une ligne peut éventuellement comporter plusieurs instructions séparées par :.

**Algobox :**
Les instructions doivent être écrites sur différentes lignes ; il faut pour cela créer d'abord ces nouvelles lignes, en utilisant le boutton Nouvelle Ligne

**XCas :**
Les instructions peuvent être séparées par un retour à la ligne SHIFT ENTER. Une ligne peut contenir plusieurs instructions séparées par ;. Attention, les lignes doivent absolument se terminer avec ;. C'est un puissant logiciel de calcul formel.

## 1.2 Les variables

**Définition 3** *On appelle variable tout emplacement de la mémoire de l'ordinateur ou de la calculatrice dans lequel on stocke une information qui peut être changée. Une variable est donc constituée :*

- *d'un nom qui permet de reconnaître où elle se situe dans la mémoire de l'ordinateur ou de la calculatrice ;*
- *d'une valeur : le nombre ou plus généralement l'information stockée.*

**Remarque :**
Les variables sous Casio ou TI peuvent contenir uniquement des nombres. Sous Algobox, XCas et autres langages de programmation pour ordinateur, les variables peuvent contenir des caractères, des lettres, des chaînes de caractères.

**Syntaxe :**
Sur Casio ou TI, on écrira 3 $\boxed{\rightarrow}$ A pour stocker le nombre 3 dans la variable A. Sur TI, la touche correspondante est $\boxed{\text{STO}\blacktriangleright}$ et sur casio $\boxed{\rightarrow}$.

Sur Algobox, on déclara la nouvelle variable dans la partie VARIABLES de l'algorithme, puis ensuite on utilisera $\boxed{\text{Affecter valeur à variable}}$ dans la partie ALGORITHME.

Enfin, sur XCas, on écrira a$\boxed{:}\boxed{=}$3.

## 1.3 Exercices sur les variables

**Exercice 1 :**

a) À l'issue de l'algorithme suivant, quel nombre est stocké dans la variable A ? Dans la variable B ?

```
3 → A
4 → B
A → C
B → A
C → B
```

b) À quoi sert l'algorithme précédent ?

## 1.4 Entrées et sorties

### Commandes d'affichage

**Définition 4** *Les commandes d'affichage servent à afficher à l'écran du texte ou la valeur d'une variable.*

**Syntaxe en algorithmique :**
Afficher `a`
ou
Afficher `"texte"`

" `texte` "
*affiche le texte entre guillemets.*
*affiche la valeur de A et attend que l'utilisateur tape sur EXE pour poursuivre l'exécution du programme.*

**TI :**

Disp " `texte` " , `A`
*affiche le texte entre guillemets puis le contenu de la variable A.*

La commande Disp ("display" en anglais c'est à dire "afficher") est accessible dans le menu PRGM I/O ("Input/Output" en anglais c'est à dire "entrée/sortie").

**Algobox :**

Ajouter AFFICHER Variable a
*affiche la valeur de la variable a*
Ajouter AFFICHER Message "texte"
*affiche le message texte*

**XCas :**

print ( " `texte` " , a ) ;
*affiche le texte suivi de la valeur de la variable a.*

### Commandes d'entrée de valeurs

**Définition 5** *Les commandes d'entrée de valeurs permettent à l'algorithme de demander à l'utilisateur un nombre, un caractère ou un texte.*

**Syntaxe en algorithmique :**

```
Saisir a
```

**Casio :**

? → A
*demande à l'utilisateur d'entrer la valeur à stocker dans la variable A.*

**TI :**

Prompt A
ou
Input A
*demande à l'utilisateur d'entre une valeur pour la variable A.*

Prompt et Input sont accessibles dans le menu PRGM I/O.

À noter, sur TI, la commande Input " texte ", A affiche le texte entre guillemets et demande d'entrer la valeur de A.

**Algobox :**

Ajouter LIRE Variable a
demande à l'utilisateur d'entrer une valeur pour la variable a et attend que la valeur soit entrée.

**XCas :**

input ( " Entrer A : " , A ) ;
*demande à l'utilisateur d'entrer une valeur pour la variable A et attend que la valeur soit entrée.*

## 1.5 Exercices sur les entrées et sorties

**Exercice 1 :**

Que fait l'algorithme suivant ?

```
Saisir A
Saisir B
A*B -> C
2*(A+B) -> D
Afficher C
Afficher D
```

**Exercice 2 :**

Que fait l'algorithme suivant ?

```
Saisir D
D/2 -> R
3,14*R^2 -> A
Afficher A
```

**Exercice 3 :**

Écrire un algorithme qui demande d'entrer deux nombres entiers A et B et calcule le reste de la division euclidienne de A et B. On utilisera pour cela la fonction partie entière `int A` qui donne la partie entière d'un nombre $a$ (menu MATH NUM iPart sur TI, menu OPTN NUM Int sur Casio et iPart sur XCAS).

**Exercice 4 :**

Écrire un algorithme qui demande d'entrer un nombre puis affiche son image par la fonction $f$ définie par $f(x) = 3x^2 + 5x - 9$.

**Exercice 5 :**

1. Écrire un algorithme qui convertit des secondes en heures, minutes et secondes.
2. Écrire un algorithme qui convertit des heures en jours et heures.

**Exercice 6 :**

Écrire un algorithme qui demande d'entrer trois nombres A, B et C et calcule et affiche leur moyenne non pondérée.

**Exercice 7 :**

Écrire un algorithme qui, l'utilisateur ayant entré le taux annuel d'épargne en pourcentage et le capital initialement placé, calcule et affiche le capital disponible auquel sont ajoutés les intérêts de l'année.

## 1.6 Structures conditionnelles

### Si..alors..sinon

**Définition 6** *Ces instructions permettent de tester si une condition est vraie ou fausse et de poursuivre le programme d'une manière différente selon que la condition est vraie ou fausse.*

**Syntaxe en algorithmique :**

Si
`condition`
Alors
`instructions si condition vraie`
Sinon
`instructions si condition fausse`
FinSi

| **TI :** | **Casio :** | **XCas :** | **Algobox :** |
|---|---|---|---|
| If `condition`<br>Then<br>`instructions`<br>Else<br>`instructions`<br>End<br><br>*If*, *Then*, *Else* et *End* sont accessibles dans le menu *PRGM* *CTL* (contrôle). | If `condition`<br>Then<br>`instructions`<br>Else<br>`instructions`<br>ifEnd<br><br>*If*, *then*, *Else* et *IfEnd* sont accessibles dans le menu *SHIFT* *PRGM* puis *COM* (*F1* sur *Graph25*). | if<br>( `condition` )<br>{ `instruction` ;<br>...<br>`instruction` ; }<br>else<br>{ `instruction` ;<br>...<br>`instruction` ; } ; | Ajouter SI...ALORS<br>`condition`<br>*si nécessaire, cocher la case SINON*<br>...<br>`Ecrire les instructions si condition vraie entre DEBUT SI et FIN SI`<br>...<br>`Ecrire les instructions si condition fausse entre DEBUT SINON et FIN SINON`<br>... |

**Exemple :**

**TI :**

```
input "A =? ",A
If A<7
Then
A+1→A
Else A-1→A
End Disp A
```

**Casio :**

```
?→A
If A<7
Then
A+1→A
Else
A-1→A
ifEnd "A"
```

**XCas :**

```
input(" a = ? ",a);
if (a<7)
    {a:=a+1;}
else
    {a:=a-1;};
print("a = ",a);
```

**Algobox :**

Ce programme teste si la variable $a$ entrée a une valeur inférieure à 7 et, si c'est le cas, ajoute 1. Sinon, il enlève 1 à la valeur de la variable. Puis, quelle que soit la valeur de $a$, il affiche le contenu de la variable $a$. On remarquera les espaces laissés au début de certaines lignes pour Xcas : ils ne sont pas indispensables mais aident à clarifier la compréhension du programme, c'est donc une bonne habitude à prendre que de les utiliser.

### Opérateurs relationnels et logiques

**Définition 7** *Pour tester une condition on utilise les* opérateurs relationnels *suivants :*
- $a = b$ *teste l'égalité de* $a$ *et de* $b$ *;*
- $a < b$ *teste si* $a$ *est strictement inférieur à* $b$ *;*
- $a \leq b$ *teste si* $a$ *est inférieur ou égal à* $b$ *;*
- $a > b$ *teste si* $a$ *est strictement supérieur à* $b$ *;*
- $a \geq b$ *teste si* $a$ *est supérieur ou égal à* $b$ *;*
- $a \neq b$ *teste si* $a$ *est différent de* $b$.

*On utiliser ausi pour les conditions plus complexes les opérateurs logiques "et" ("AND"), "ou"("OR") et "non" ("not").*

**TI :**
Les opérateurs relationnels se trouvent dans 2nd TEST TEST et les opérateurs logiques dans LOGIC.

**XCas :**
- $a == b$ teste l'égalité de $a$ et de $b$ ;
- $a < b$ teste si $a$ est strictement inférieur à $b$ ;
- $a <= b$ teste si $a$ est inférieur ou égal à $b$ ;
- $a > b$ teste si $a$ est strictement supérieur à $b$ ;
- $a >= b$ teste si $a$ est supérieur ou égal à $b$ ;
- $a! = b$ teste si $a$ est différent de $b$ ;
- `condition1 && condition2` teste si les deux conditions sont vraies simultanément ;
- `condition1 || condition2` test si l'une au moins des deux conditions est vraie ;
- `!condition` teste si la négation de la condition est vraie.

**Algobox :**
- $a == b$ teste l'égalité de $a$ et de $b$ ;
- $a < b$ teste si $a$ est strictement inférieur à $b$ ;
- $a <= b$ teste si $a$ est inférieur ou égal à $b$ ;
- $a > b$ teste si $a$ est strictement supérieur à $b$ ;
- $a >= b$ teste si $a$ est supérieur ou égal à $b$ ;
- $a! = b$ teste si $a$ est différent de $b$ ;
- `condition1 ET condition2` teste si les deux conditions sont vraies simultanément ;
- `condition1 OU condition2` test si l'une au moins des deux conditions est vraie ;

## 1.7 Exercices sur les structures conditionnelles

**Exercice 1 :**

Écrire un programme qui demande l'âge de l'utilisateur et répond "vous êtes mineur" ou "vous êtes majeur" suivant le cas.

**Exercice 2 :**

Écrire un programme qui demande la température extérieure en degrés celsius et affiche "il gèle" si le nombre est négatif et "alerte à la canicule" si le nombre est supérieur à 30.

**Exercice 3 :**

1. Qu'affiche l'algorithme suivant ?

```
1000->tirelire
19->âge
Si (âge >=19 et tirelire >=1000)
alors afficher "Vous pouvez ouvrir un compte"
sinon afficher "pas de compte possible"
```

2. Écrire le code correspondant à l'algorithme précédent pour la calculatrice ou pour XCas.

**Exercice 5 :**

Écrire un algorithme qui, à partir d'un nombre entré par l'utilisateur, affiche ce même nombre s'il est positif et son opposé s'il est négatif (le nombre obtenu est appelé la valeur absolue du nombre entré).

**Exercice 6 :**

Écrire un algorithme qui, à partir de la donnée de la longueur de chacun des trois côtés d'un triangle, teste si le triangle est rectangle.

## 1.8 Boucles

**Définition 8** *Les boucles sont utilisées pour qu'une séquence d'instructions soit répétée un nombre donné de fois ou tant qu'une condition n'est pas remplie.*

### Boucles "pour"

**Définition :**

Ces instructions sont utilisées pour contrôler les boucles en incrémentant (augmentant) une variable. La variable est augmentée d'une valeur de départ jusqu'à une valeur d'arrivée d'un pas donné (l'incrément).

**Syntaxe :**
Pour
`variable`
allant de
`valeur de départ`
à
`valeur d'arrivée`
faire
`instructions`
fin

**Casio :**

For
`valeur de départ→variable` To `valeur d'arrivée` Step `incrément` `instructions`
Next

*Les instructions For, To, Step, Next se trouvent dans SHIFT PRGM COM.*

**TI :**

For ( `variable` , `valeur de départ` , `valeur d'arrivée` , `incrément` )
`instructions`
End

*Les instructions For, End se trouvent dans le menu PRGM CTL.*

**XCas :**

```
For ( variable := valeur de départ ;
          variable := valeur finale ; variable := variable + incrément )
{ instruction ;
instruction ;
...
instruction ; } ;
```

**Algobox :**

```
Ajouter POUR...DE...A variable allant de valeur de départ à valeur finale
L'incrémentation se fait automatiquement de 1 en 1.
```

**Exemple :**

```
Pour a allant de 0 à 10 par pas de 2 faire
   a*a -> b
   Afficher a et b
Fin Pour
```

**Casio :**

```
For 0 → A To 10 Step 2
A * A → B
B
Next
```

**TI :**

```
For ( A , 0 , 10 , 2 )
A * A → B
Disp A , B
End
```

**XCas :**

```
for (a:=0; a<=10;a:=a+2)
   {b:=a^2;
   print(a,b);}
```

**Algobox :**

```
1   VARIABLES
2     A EST_DU_TYPE NOMBRE
3     B EST_DU_TYPE NOMBRE
4     C EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     POUR A ALLANT_DE 0 A 5
7       DEBUT_POUR
8       C PREND_LA_VALEUR 2*A
9       B PREND_LA_VALEUR C*C
10      AFFICHER C
11      AFFICHER B
12      FIN_POUR
13  FIN_ALGORITHME
```

Cet algorithme affiche le tableau de valeurs de la fonction carré de 0 à 10 par pas de 2.

### Boucles "Tant que"

**Définition 9** *Éxécute un groupe de commandes tant qu'une condition est vraie. La condition est testée en début de boucle.*

**Syntaxe :**
Tant que `condition`
`instructions`
faire
`instructions`
fin tant que

**Casio :**
While `condition`
`instructions`
WhileEnd

*While* et *WhileEnd* se trouvent dans le menu *SHIFT* *PRGM* *COM* (*F1* sur *Graph25*)

**TI :**
While `condition`
`instructions`
End

*While* et *End* se trouvent dans le menu *PRGM* *CTL*.

**XCas :**
While ( `condition` )
{ `instruction` ;
`instruction` ;
...
`instruction` ; }

**Algobox :**
Ajouter TANT QUE
`condition`
DEBUT TANT QUE
`instruction1 instruction2`
...
FIN TANT QUE

**Exemple :**

```
10 -> a
Tant que a>0 faire
   a-1 -> a
   Afficher a
fin tant que
```

**TI :**

$10 \to A$
While $A > 0$
$A - 1 \to A$
Disp $A$
End

**Casio :**

$10 \to A$
While $A > 0$
$A - 1 \to A$
WhileEnd

**XCas :**

```
a:=10;
while (a>0)
   {a:=a-1;
   print(a);};
```

**Algobox :**

```
1   VARIABLES
2     A EST_DU_TYPE NOMBRE
3   DEBUT_ALGORITHME
4     A PREND_LA_VALEUR 10
5     TANT_QUE (A>0) FAIRE
6       DEBUT_TANT_QUE
7       A PREND_LA_VALEUR A-1
8       AFFICHER A
9       FIN_TANT_QUE
10  FIN_ALGORITHME
```

Cet algorithme affiche le décompte de 9 à 0.

**Exemple :**

```
Saisir A
Saisir B
1->R
Tant que R<>0
faire
  A-B*Int(A/B) -> R
  Afficher R
  B->A
  R->B
FinTantque
Afficher "PGCD=",A
```

**TI :**

```
Input "A=","A
Input "B=","B
1→ R
While R ≠ 0
A-B*iPart(A/B)→ R
Disp R
B→A :R→B
End
Disp "PGCD=","A
```

**Casio :**

```
"A=" ?→A
"B=" ?→B
1→ R
While R≠0
A-B*Int(A/B)→ R
B→A :R→B
WhileEnd
```

**XCas :**

```
input("a= ",a);
input("b= ",b);
r:=1;
while (r!=0) {
r:=a-b*intDiv(a,b);
print("r= ",r);
a:=b;b:=r;
};
print("PGCD =",a);
```

**Algobox :**

```
1   VARIABLES
2     A EST_DU_TYPE NOMBRE
3     B EST_DU_TYPE NOMBRE
4     R EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     LIRE A
7     LIRE B
8     R PREND_LA_VALEUR 1
9     TANT_QUE (R!=0) FAIRE
10      DEBUT_TANT_QUE
11      R PREND_LA_VALEUR A-B*floor(A/B)
12      AFFICHER R
13      A PREND_LA_VALEUR B
14      B PREND_LA_VALEUR R
15      FIN_TANT_QUE
16    AFFICHER "PGCD = "
17    AFFICHER A
18  FIN_ALGORITHME
```

Cet algorithme utilise l'algorithme d'Euclide pour calculer le PGCD de deux entiers $A$ et $B$ entrés.

### Boucles "répéter"

**Définition 10** *Comme les boucles "tant que", une boucle "répéter" éxécute un groupe d'instructions mais ceci jusqu'à ce que la condition soit vraie et la condition est testée en fin de boucle. Dans les deux cas, la boucle est toujours réalisée au moins une fois.*

**Syntaxe :**

```
Répéter
instructions
jusqu'à condition
```

| **Casio :** | **TI :** | **XCAS :** | **Algobox :** |
|---|---|---|---|
| Do<br>instructions<br>LpWhile condition<br><br>*Do* et *LpWhile* (”Loop” en anglais signifie ”boucle”) se trouvent dans le menu *SHIFT* *PRGM* *COM* (*F1* sur *Graph25*).<br>*Attention sur Casio, la boucle Do LpWhile s'effectue tant que la condition est vraie et non pas jusqu'à ce que la condition soit vraie.* | repeat condition<br>instructions<br>End<br><br>*repeat* et *End* se trouvent dans le menu *PRGM* *CTL*.<br>*Attention sur TI : La condition se met au début de la boucle mais elle est testée en fin de boucle uniquement.* | *Pas de telle boucle pour XCAS, utiliser une boucle ”tant que”.* | *Pas de telle boucle pour Algobox, utiliser une boucle ”tant que”.* |

**Exemple :**

```
Saisir A
Saisir B
1->R
Répéter
  A-B*Int(A/B) -> R
  Afficher R
  B->A
  R->B
jusqu'à R=0
Afficher "PGCD=",A
```

| **TI :** | **Casio :** |
|---|---|
| Input ”A=”,A<br>Input ”B=”,B<br>1→ R<br>Repeat R = 0<br>A-B*iPart(A/B)→ R<br>Disp R<br>B→A : R→B<br>End<br>”PGCD=”,A | ”A=” ?→A<br>”B=” ?→B<br>Do<br>A-B*Int(A/B)→ R<br>B→A : R→B<br>LpWhile R ≠ 0 |

Il s'agit du même algorithme d'Euclide. Observer les différences avec l'algorithme écrit à l'aide de boucles ”tant que” et les différences d'écriture sur les modèles TI et Casio.

## 1.9 Exercices sur les boucles

**Exercice 1 :**

1. Combien de fois le message "Salut" sera-t-il affiché à partir de l'algorithme suivant ?

```
15 -> A
Répéter
  afficher "Salut"
  A+1->A
jusqu'à A<15
```

2. Combien de fois ce même message sera-t-il affiché dans le cas suivant ?

```
14->A
Tant que A<15
faire
  afficher "Salut"
finTantque
```

**Exercice 2 :**

1. Écrire un algorithme qui calcule la somme des nombres entiers de 0 à 50.
2. Écrire un algorithme qui calcule le produit des nombres entiers de 1 à 7
3. Écrire un algorithme qui calcule la somme des 20 premiers nombres impairs.
4. Écrire un algorithme qui calcule la somme des 20 premiers nombres paires.

**Exercice 3 :**

Écrire un algorithme qui calcule la variance et l'écart type d'une série de nombres entrés par l'utilisateur. L'algorithme demandera le nombre de nombres que comprend la série avant de demander d'entrer la série de nombres.

**Exercice 4 :**

Écrire un algorithme qui, une somme initiale ayant été demandée à l'utilisateur ainsi qu'une durée de placement en année et un taux de placement en pourcentage à intérêts composés, affiche la somme disponible au bout de la durée de placement.

**Exercice 5 :**

Écrire un algorithme permettant le calcul du PGCD de deux nombres entrés par l'utilisateur par la méthode des différences successives (on rappelle que les différences successives consistent à faire la différence du plus grand nombre par le plus petit et à garder la différence et le plus petit nombre à chaque étape pour recommencer jusqu'à obtention de 0).