PROGRAMMATION

Plutôt que nous lancer dans des considérations générales sur l'algorithmique, nous préférons commencer directement par de la programmation. Il sera toujours temps de disserter ensuite. Pour cette initiation, nous avons fait le choix de la calculatrice Ti-82 Stats.

I- MISE EN ROUTE

MATÉRIEL

On va supposer que vous avez devant vous une TI-82 Stats.fr éteinte, mais en état de fonctionner, avec des piles chargées. Commencez par allumer votre calculatrice :

Vous voyez alors apparaître un joli nécran avec un curseur clignotant en haut à droite. Si ce n'est pas le cas, appuyez deux fois sur annul. Si ce n'est toujours pas le cas, éteignez

([OFF] = 2nde ON)et recommencez. Le mieux serait même de tout réinitialiser (allez dans [mém] = 2nde +). Si ça ne fonctionne toujours pas, tant pis. Si ça fonctionne, eh bien :

BRAVO

ALGORITHME

Dans les deux cas, en suivant à la lettre les instructions, vous avez exécuté un « programme », ou un « algorithme ». C'est déjà une première façon de prendre contact avec le sujet.

II- AFFECTATION (et calcul)

HOMONYMIE

On entend le mot d'*affectation* au sens d'attribution (« affectation d'un budget à l'armée ») et non au sens d'hypocrisie, de minauderie (« affectation dans les manières »).

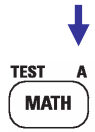
VARIABLES

Les lettres en vert (pour les calculatrices noires, mais en orange pour les violettes) en haut à droite de la plupart des touches, désignent des « *mémoires* » de la calculatrice, qui peuvent contenir des nombres. Elles ont le rôle des *variables* en mathématiques (et nous les appellerons indifféremment *variables* ou *mémoires*), à ceci

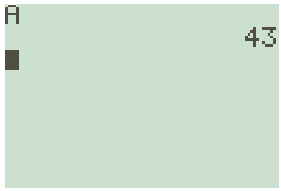
près qu'elles ne peuvent pas toujours contenir une valeur exacte (une précision infinie demanderait une mémoire infinie), mais, parfois, seulement une valeur approchée. En outre, les valeurs trop grandes (à partir de 10^{100}) dépassent les capacités de la mémoire.

On accède aux variables grâce à la touche [alpha].

Pour afficher « A », tapez [alpha] puis la touche :



Pour connaître la valeur contenue actuellement dans la mémoire A de votre calculatrice, tapez « A [entrer] ».



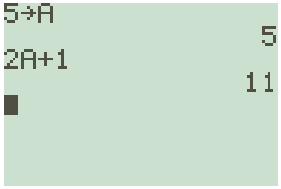
Ici, la calculette affiche 43, mais la vôtre peut aussi bien afficher n'importe quelle autre valeur. Par défaut (après réinitialisation), les variables sont toutes remises à 0.

Store

Pour donner à A une valeur de votre choix, vous pouvez utiliser la touche [sto→]. « Sto » est l'abréviation de « *to store* », qui veut dire « emmagasiner », en anglais. Nous l'écrirons comme la machine l'affiche « → » et la lirons « dans ». Pour mettre la valeur 5 dans la mémoire A, tapez « $5 \rightarrow A$ ». L'ancienne valeur de A est écrasée par la nouvelle : la machine n'en a plus le souvenir.

Vous pouvez ensuite non seulement redemander la valeur de A à

la machine (même si elle a été éteinte entre temps), mais surtout, faire des calculs avec cette variable. Par exemple, si vous tapez « $2A+1$ [entrer] », la machine calcule la valeur de l'expression « $2A+1$ » avec la valeur actuelle de A (qui est 5). Il est donc logique qu'elle affiche…



Il peut y avoir plusieurs variables dans une expression que vous faites calculer. Exemple : « $3A + B^2$ ».

Syntaxe

À droite de « → », il doit toujours y avoir une variable. (Ici, si vous pensez à la flèche à talon des fonctions, oubliez. D'abord, c'est à gauche de la flèche à talon qu'il y a toujours une variable et puis les deux flèches ne signifient vraiment pas la même chose.) À gauche, en revanche, on peut mettre une expression. Exemple : « $2A+1 \rightarrow B$ ». Lorsque vous appuierez sur [entrer], la machine calculera la valeur de l'expression située à gauche de la flèche, la stockera dans une mémoire provisoire intermédiaire, puis mettra cette valeur dans la mémoire B ? Pourquoi est-ce que je complique en vous parlant de mémoire intermédiaire ? Tout simplement pour que vous compreniez qu'on peut écrire : « $2A+1 \rightarrow A$ ».

Instruction

Une « phrase » comme « $2A+1 \rightarrow B$ » n'est ni une *affirmation* ni une *expression* : elle n'est ni vraie ni fausse et n'a pas de valeur. C'est un ordre. Il faudrait utiliser l'impératif pour en rendre

précisément le sens : « calcule $2A+1$ et mets le résultat dans B ».

Une ordre donné à la machine, qu'elle exécutera servilement, c'est-à-dire stupidement (Hegel, Heidegger et les frères Wachowski nous disent de nous méfier, mais passons), s'appelle une **instruction**. Une instruction fondée sur « → » est une instruction d'**affectation**.

Répétition

Lorsque, juste après avoir fait faire un calcul par la machine, on « rappuye » sur la touche [entrer], la machine exécute à nouveau le dernier calcul.

Essayez par exemple : « $2A+1 \rightarrow A$ [entrer] [entrer] [entrer]… »

Amusant, non ?

Traductions

TI82 Stats.fr, Casio	$5 \rightarrow A$
Basic, C, Python :	$A = 5$
Pascal	$A := 5$;
Langages « pédagogiques »	A prend la valeur 5
	Affecter à A la valeur 5

III- Entrée /sortie

Programme

Un **programme** est une succession d'*instructions* que la machine n'exécute pas tout de suite, mais qu'elle exécutera les unes après les autres lorsqu'on le lui demandera ; lorsqu'on

« lancera » le programme. On peut lancer un programme autant de fois qu'on veut sans avoir à le récrire.

Le mot *programme* est composé de *pro-*, avant (comme dans prologue) et de *gramma* « ce qui est écrit » (comme dans grammaire). Littéralement : ce qui est écrit à l'avance.

Pour vous mettre en mode *Programme*, appuyez sur la touche [prgm]. Ensuite, pour créer un nouveau programme, déplacez le curseur sur « NOUV » (Nouveau), puis appuyez sur [entrer]. Tapez ensuite le nom que vous voulez donner à votre programme, peut importe lequel, par exemple « ESSAI » (n'appuyez pas sur [alpha] pour accéder aux lettres). Puis [entrer].

Les deux points (« : ») en début de ligne signalent que la machine attend vos instructions. Elle ne les exécutera pas tout de suite, mais les mémorisera sagement. Nous allons taper le programme suivant :

```
15 → A
2A+1 → A
Disp A
```

Pour obtenir l'instruction « Disp », tapez d'abord [prgm]. Comme vous êtes déjà en mode *Programme*, cela vous conduira à un menu donnant accès à toutes les instructions. Il y a trois sous-menus, CTL (Instructions de Contrôle), E/S (Instructions d'Entrée/Sortie) et EXEC (qui donne accès aux programmes

eux-mêmes). L'instruction « Disp » est dans le sous-menu E/S. Nous allons en expliquer le sens.

Display

To Display veut dire « afficher », en anglais. Cette instruction donne l'ordre à la machine d'afficher la valeur de l'expression qui la suit. L' expression en question peut être réduite au strict minimum (variable ou constante). On peut aussi faire afficher un texte, à condition de le mettre entre guillemets. Par exemple : Disp "COUCOU" (pour taper plusieurs lettres à la suite : [verr A] = (2nde)(alpha) ; les guillemets sont au dessus de la touche [+]). *Disp* est une instruction de « sortie » : elle permet à la machine de faire sortir de l'information (par le biais de l'écran) pendant le déroulement du programme.

Dans le mode de calcul classique, nous n'avions pas besoin de l'instruction « Disp » : le seul fait de taper une expression, puis (entrer), conduisait la machine à calculer puis afficher la valeur de cette expression. En mode *Programme*, c'est différent.

Lancement

Avant que nous ne lancions le programme, essayez d'en anticiper le résultat : mettez-vous à la place de la machine et exécutez les trois instructions les unes après les autres. Vous pouvez, sur une feuille de brouillon, dessiner un cadre dans lequel vous inscrirez les valeurs successives de *A* (lorsque vous marquez une nouvelle valeur, barrez la précédente) et un autre qui représentera l'écran.

À présent, nous allons lancer le programme. Revenez en mode de calcul classique avec [quitter] ((2nde) (mode)). Puis (prgm), pour retourner en mode programme. Allez ensuite sur EXEC, puis (entrer). Choisissez votre programme (flèches directionnelles pour

déplacer le curseur), puis (entrer) (entrer). C'est un peu compliqué, mais je n'en suis pas responsable.

Alors ? Aviez-vous prévu ce qui arrive ?

À la fin, la machine affiche « Fait » pour signaler que le programme est arrivé à son terme.

Traductions

TI82 Stats.fr	Disp *A*
Casio	*A*◢
Basic	Print *A*
Python	Print(*A*)
Pascal	WriteLn(*A*) ;
Langages « pédagogiques »	Afficher *A*

Entrée

Avec les instructions dont nous disposons jusqu'ici, un programme fera toujours la même chose : aucune intervention extérieur (ni aucun hasard, soit dit en passant) ne pourra modifier sa course aveugle. Nous allons voir comment, lors de l'exécution du programme, la machine peut recevoir des informations du monde extérieur, qui pourront intervenir dans son déroulement.

Input

Input est une instruction d'entrée. Pour vous faire comprendre cette instruction, nous allons distinguer trois « personnes » : le programmeur, la machine et l'utilisateur. L'utilisateur, c'est celui qui sera devant la machine pendant l'exécution du programme. Dans la pratique, il est possible que ce soit le même individu que le programmeur (vous, par exemple), mais dans des rôles bien différents. Alors maintenant, concentration :

Par l'instruction « Input *A* », le programmeur donne l'ordre à la machine de demander à l'utilisateur de saisir une valeur. La machine doit ensuite mettre cette valeur dans la mémoire *A*.

Qu'est-ce que vous n'avez pas saisi ? Le mot *saisir* ? Ça veut dire taper le nombre au clavier et appuyer sur [entrer]. Comment la machine demande à l'utilisateur d'entrer une valeur ? Elle affiche juste un point d'interrogation à l'écran. J'essaye de redire les choses autrement :

Lorsque la machine rencontre l'instruction « Input *A* » pendant l'exécution du programme, d'abord elle affiche un point d'interrogation à l'écran et attend. Attend quoi ? Que l'utilisateur veuille bien saisir un nombre (par exemple, il tape « 23[entrer] »). Cette valeur est ensuite emmagasinée dans la mémoire *A* (ou autre si l'on a indiqué une autre variable).

Vous n'avez toujours pas compris ? Ben relisez. La pédagogie, ça va deux minutes. À vous de réfléchir, un peu. On va écrire un petit programme pour faire fonctionner *Input*, ça va peut-être vous aider :

```
Input A
2A+1 → A
Disp A
```

Astuces

Inutile de tout taper à nouveau : vous pouvez juste modifier le programme précédent. Allez dans le programme que vous venez d'écrire : pour cela, dans le mode *Programme*, allez dans le menu *Edit* (*éditer* un programme, c'est le modifier). Il vous suffit de modifier la première ligne. Vous pouvez utiliser [suppr] et aussi [insérer] (=[2nde] [suppr]). Pour relancer un programme qui vient d'être exécuté, appuyez sur [entrer].

Pour effacer un programme, allez dans [mém] (= [2nde] [+]).

Syntaxe

Après *Input*, il doit toujours y avoir une variable.

Erreur courante

Parfois, certains étourdis confondent *Input* et *Sto→*. Il est vrai que dans les deux cas, on met une valeur dans une variable. Mais ce n'est pas la même « personne » qui le fait.

Traductions

TI82 Stats.fr, Basic	Input *A*
Casio	? → *A*
Python	A = Input()
Pascal	ReadLn(*A*) ;

Langages « pédagogiques »

Saisir *A*

Demander à l'utilisateur la valeur de *A*

IV- Saut

Principe

Pour l'instant, les instructions du programme sont exécutées dans l'ordre dans lequel elles sont écrites. Mais on pourrait vouloir, par exemple, repartir au début du programme une fois arrivé à la fin (sans que l'utilisateur ait besoin d'appuyer sur entrer). Les instructions qui suivent permettent au programme de modifier lui-même son déroulement…

Goto, Lbl

Label signifie « étiquette » en anglais, et *to go to* signifie « aller à ». Voyons cela sur un exemple :

Lbl 1
Input *A*
$2A+1 \rightarrow A$
Disp *A*
Goto 1

Ajoutons donc deux lignes (en gras ci-dessus) au programme précédent. Pour insérer une ligne, placez le curseur au début de la ligne avant laquelle vous voulez en insérer une autre, puis : [insérer] (entrer).

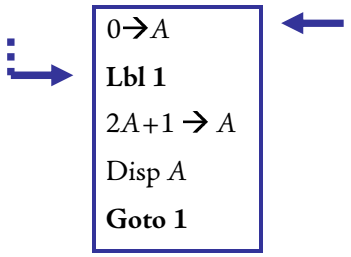
Lorsque la machine rencontre l'instruction « Lbl 1 »… elle ne fait rien de particulier. Elle se contente de poursuivre en passant à l'instruction suivante. L'instruction *Lbl* n'en est pas vraiment une : elle sert juste à marquer un endroit du programme.

Lorsque la machine rencontre l'instruction « Goto 1 », elle va à l'endroit où se trouve l'étiquette 1 (peu importe que celle-ci soit située avant ou après le Goto) et exécute l'instruction suivante (puis encore les suivantes, s'il n'y a pas de nouveau saut). Le

numéro choisi ne joue aucun rôle en soi. Si vous remplacez « Lbl 1 » et « Goto 1 » par « Lbl 42 » et « Goto 42 », cela reviendra au même. L'étiquette, sur la Ti82 Stats, peut aussi contenir des lettres.

Lancez le programme, observez son exécution.

Nous allons ensuite lui apporter un légère modification, en déplaçant l'étiquette. Nous allons aussi remplacer le *Input* par un Sto→, afin qu'il vous soit plus facile d'imaginer ce que va faire la machine :



Essayez, là encore, d'anticiper l'action de la calculatrice : mettez-vous à sa place en exécutant les instructions du programme le plus bêtement possible.

Astuces

Pour arrêter un programme en cours, appuyez sur [ON].

Si vous voulez avoir le temps de voir chaque nombre affiché, vous pouvez ajouter l'instruction *Pause*, après de *Disp* :

```
0→A
Lbl 1
2A+1 → A
Disp A
Pause
Goto 1
```

Lorsqu'elle rencontre cette instruction, la machine attend que vous appuyiez sur (entrer) avant de poursuivre.

Traductions

TI82 Stats.fr, Casio	Goto 1	Lbl 1
Basic (ancienne version)	Goto <numéro de ligne>	

En Pascal, Python et dans tous les langages où l'on prétend faire de la programmation « structurée », on à tendance à éviter les sauts. Certains puristes voudraient les y interdire totalement. (Les sauts sont alors remplacés par d'autres instructions, notamment par des instructions de « branchement » qui sont des « appels à des sous-programmes ».)

V- Condition

Problématique

Le programme précédent ne s'arrête que parce qu'à un moment, les capacités de la calculatrice sont dépassées (la valeur de *A* devient trop grande) ou parce que l'utilisateur interrompt le programme. Il serait intéressant que la boucle s'arrête toute seule à partir d'une certaine valeur de *A*. Pour cela, il faut que l'intruction *Goto* ne soit exécutée qu'à la condition que *A* soit inférieur à une valeur donnée.

« If— »

If veut dire « si », en anglais. « *If*— » est aussi le titre d'un poème fort célèbre de Rudyard Kipling, écrit en 1895, et publié en 1910… Mais revenons à nos considérations prosaïques. Cette instruction permet de ne faire exécuter l'instruction qui la suit que si une certaine **condition** est vérifiée.

Une *condition* est tout simplement une *affirmation*, c'est-à-dire une phrase qui est soit *vraie*, soit *fausse*.

```
0→A
Lbl 1
2A+1 → A
Disp A
If A<1000
Goto 1
Disp "FINI"
```

Pour accéder au signe « < », allez dans [tests] (= (2nde)(math)).

En lançant le programme, vous pourrez constater que la dernière valeur affichée est supérieur à 1000. Voyez-vous pourquoi ?

TRADUCTIONS

TI82 Stats.fr, Casio	If <Condition> <Instruction>
PureBasic	If <Condition> <InstructionS> EndIf
Langages « pédagogiques »	Si *A*>100 alors <Instruction>

REMARQUES Lorsqu'on utilise l'instruction *If*, seule la ligne suivante est exécutée sous condition. Le programme poursuit ensuite sa course normalement. On peut considérer, que la ligne commençant par *If* et la suivante n'en forment qu'une seule.

Il existe une façon de rendre plusieurs lignes successives conditionnelles, mais pour l'instant, nous nous contentons du minimum.

La ligne qui suit le « *If*… » peut contenir n'importe quelle instruction et pas forcément un *Goto*.

VI- RÉCAPITULATION

RÈGLE DU JEU

À présent, vous disposez de toutes les instructions nécessaires, qui sont les briques à partir desquelles vous allez construire vos programmes, vos **algorithmes**. On pourrait encore voir une ou deux instruction d'affichage (et nous les verrons plus loin),

mais en dehors de cela, de nouvelles instructions ne pourraient vous apporter qu'un peu de confort, seraient des raccourcis, mais n'offriraient aucune possibilité nouvelle. Dans un premier temps, vous allez vous contenter de ce qui a été vu jusqu'ici. Ne cherchez pas à inventer de nouvelles instructions et n'attendez pas que la machine comprenne vos intentions : elle exécute bêtement des ordres très simples. L'intelligence et la créativité sont de votre seul ressort.

LES 5 INSTRUCTIONS ÉLÉMENTAIRES

	Exemple	Mot anglais	Traduction	**Ordre donné à la machine**
Affectation (et calcul)	A+1→B Touche « sto→ » sera lue « dans »	To store	Emmagasiner	Calcule la valeur de l'expression de gauche, puis mets-la dans la variable indiquée à droite. L'expression de gauche peut se réduire à une constante ou une variable : 5→A ; A → B. A droite, il y a toujours une variable. Ainsi, l'instruction « 5 → A+1 » ne veut rien dire.
Entrée	**Input** A	To put in	Mettre dans	Demande à l'utilisateur de saisir une valeur, puis affecte-la à la variable indiquée.
Sortie	**Disp** A	To display	Afficher	Affiche la valeur actuelle de la variable indiquée. On peut aussi demander l'affichage de la valeur d'une expression : Disp 2A+1 On peut aussi faire afficher un texte, on utilise alors des guillemets : Disp « coucou »
Saut	Lbl 2	Label	Étiquette	Ne fait rien, ceci sert juste à marquer endroit du programme. Le « 2 » dans l'exemple est juste le numéro de l'étiquette. On peut le remplacer par n'importe quel autre numéro.
	Goto 2	To go to	Aller à	Au lieu de passer à l'instruction suivante, va à l'instruction qui suit l'étiquette indiquée.
Condition	**If** A<100 Goto 1	If	Si	N'exécute l'instruction de la ligne suivante que si la condition est vérifiée (sinon, passe à l'instruction d'après). À la place du « Goto 1 » de l'exemple, n'importe quelle autre instruction est acceptée.

VII- INSTRUCTIONS COMPLÉMENTAIRES

Au risque de me répéter, je vous recommande, dans un premier temps, de vous contenter des 5 instructions élémentaires. Par la suite, vous pourrez utiliser ce qui suit si vous en ressentez le besoin.

IF … THEN … ELSE

Ti 82 Stats

```
If <Condition>
Then
<InstructionS (1)>
[Else
<InstructionS (2)>]
End
```

Équivaut à :

```
If <Condition>
Goto 1
<InstructionS (2)>
Goto 2
Lbl 1
<Instructions (1)>
Lbl 2
```

Languages « pédagogiques »

```
Si <Condition> alors
    <InstructionS>
Sinon
    <InstructionS>
```

PureBasic :

```
If <Condition>
<InstructionS>
EndIf
```

While

Ti 82 Stats	Languages « pédagogiques »
While <Condition> <InstructionS> End	Tant que <Condition> └ <Instructions>
Équivaut à :	PureBasic :
Lbl 1 If <Condition> Then <InstructionS> Goto 1 End	While <Condition> <InstructionS> Wend

Repeat

Ti 82 Stats	Languages « pédagogiques »
Repeat <Condition> <InstructionS> End	Répète └ <InstructionS> Jusqu'à ce que <Condition>
Équivaut à :	PureBasic :
Lbl 1 <InstructionS> If non(<Condition>) Goto 1	Repeat <InstructionS> Until <Condition>

Remarques

Repeat est proche de While. La différence importante étant qu'avec Repeat, la boucle est toujours effectuée au moins une fois. Dans les textes plus ou moins officiels (commentaires des programmes, énoncés du bac), *While* est préféré à *Repeat*.

Il faut comprendre le « Repeat <Condition> » de la Ti82 comme un « Repeat (until) <Condition> » : répète jusqu'à ce que la condition soit vérifiée.

For

Ti 82 Stats	Languages « pédagogiques »
For(A,1,10) <InstructionS> End	Pour A de 1 à 10 └ <Instructions>
Équivaut à :	PureBasic
$1 \rightarrow A$ While $A \leq 10$ <Instructions> $A+1 \rightarrow A$ End	For i=1 to 10 <InstructionS> Next i

Remarques

À la place des constantes « 1 » et « 10 » de l'exemple, on peut mettre des variables ou des expressions.

While, *Repeat* et *For* servent à faire des **boucles**.

VIII- ÉCRAN GRAPHIQUE

PIXELS

Un **écran graphique** peut être vu comme un quadrillage dont chaque petit carreau, qui s'appelle un **pixel**, peut s'allumer et s'éteindre (ou se colorier) indépendamment des autres. Le pixel est grossièrement l'équivalent du point, en géométrie, à ceci près qu'il y a une infinité de points dans le plan, dans un rectangle donné et même sur un segment, mais qu'il n'y a qu'un nombre limité de pixels sur un écran.

L'écran de la Ti82, un peu rudimentaire, a une dimension de 95 pixels en largeur et 63 pixels en hauteur ; et chacun de ces pixels n'a que deux états : « allumé » (colorié en noir) ou « éteint ».

PRÉCAUTIONS

Sur la Ti82, un mauvais choix dans les paramètres peut générer des erreurs inattendues. Nous allons donc faire un peu le ménage.

(mode) : griser toutes les options de gauche, notamment Fct.

(f(x)) : avec (annul), effacer toutes les expressions de fonctions et avec (entrer), « dégriser » Graph1, Graph2, Graph3 (en haut de l'écran).

[format] (=(2nde)(zoom)) : on va sélectionner toutes les options de gauche, sauf AxesAff : pour les axes on choisira de ne pas les afficher en sélectionnant AxesNAff à droite.

(fenêtre) : on choisit Xmin=0 ; Xmax=94 ; Xgrad=1 ; Ymin=0 ; Ymax=62 ; Ygrad=1 ; Xrés=1.

Si vous voulez des précisions sur toutes ces options, il faudra peut-être songer à aller voir la notice de votre machine, téléchargeable gratuitement sur le site Texas.

INSTRUCTIONS

Dans le menu [dessin] (=(2nde)(prgm)), nous trouvons les instructions concernant le dessin. Nous n'en utiliserons que trois : EffDessin, Pt-On() et Ligne(). Deuxième est dans le sous-menu « POINTS », les autres dans le sous-menu « DESSIN ».

EFFACE DESSIN

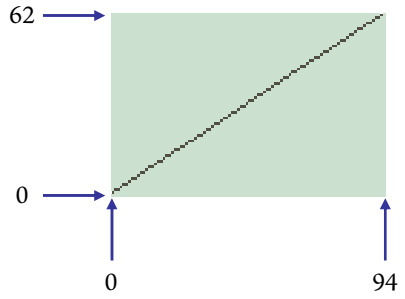
EffDessin efface l'écran graphique. Vous avez intérêt à mettre cet instruction au début de tout programme qui utilise l'écran graphique.

POINT ON

Pt-On(20,3) : affiche le point (le pixel) d'abscisse 20 et d'ordonnée 3. Attention, les coordonnées sont séparées par une virgule (touche [,]) et non, comme en mathématiques, par un point-virgule.

LIGNE

Ligne(0,0,94,62) trace la ligne qui va du point de coordonnées (0 ; 0) à celui de coordonnées (94 ; 62) :



REMARQUE Les paramètres des instructions *Point On* et *Ligne* ne sont pas forcément des constantes : elles peuvent aussi être des variables ou des expressions.