Les listes en Basic Casio

Auteur : pylaterreur
Créé : le 13/04/2008 à 09:54:41
Modifié : Aujourd'hui à 13:07:52

Salut à tous,
Dans ce tuto, vous allez apprendre à utiliser les fonctions spécifiques aux listes, utilisées dans le langage Basic Casio, c'est-à-dire le langage des calculatrices Casio Graph 20~100.
Sachez que je ne possède qu'un seul modèle de calculatrice : la Graph 85 SD (la meilleure), ce qui fait que je ne suis pas sûr à 100% que ce que je dis s'appliquera de la même façon pour une Graph 35+, par exemple. Néanmoins, ces calculatrices restent très proches au niveau de l'interprêtation du Basic Casio, donc il ne devrait pas y avoir de problème. Si problème il y avait, faites-m'en part dans les commentaires ou par MP.

Pour pouvoir suivre ce tutoriel, il est fortement recommandé d'avoir une petite expérience en Basic Casio.

Sommaire du chapitre :

- Les possibilités offertes
- Les fonctions spécifiques
- DrawStat, le dessin-éclair

Les possibilités offertes

Petit rappel : les listes sont des tableaux à une dimension, qu'on peut utiliser dans un programme, dans un calcul (Menu RUN), et dans le Menu LIST (ou Menu STAT, pour la Graph 85).
Sur la Graph 35+, elles sont au nombre de 6 (numérotées de 1 à 6) + la List Ans. Cette dernière n'est pas une liste comme les autres, elle est aux listes ce qu'Ans est aux variables. On ne peut donc pas attribuer un contenu à la List Ans avec la flèche ->.

Les listes peuvent être utilisées pour dessiner plus rapidement qu'avec des F-Line, grâce au DrawStat. Mis à part cette utilisation graphique, les listes sont déjà un outil très puissant :

- on est plus limité par le nombre de variable (qui sont au nombre de 26 + 2 = 28 : $A \sim Z$, θ et r).
- on peut faire des calculs à répétition, à l'aide d'une boucle, en se déplaçant dans la liste
- on peut faire des calculs sur toute les cases de la liste, en une seule fois

J'oublie sûrement des éléments, mais je les rajouterai au fur et à mesure du tuto.

Commençons par le plus simple, des calculs sur une liste (dans cet exemple, il s'agit d'une addition, mais vous pouvez faire ce que vous voulez, vous pouvez même envoyer une liste à une fonction comme `Abs`) :

Code : Autre - Addition d'une liste et d'un nombre

```
{5,2,3,5}
List Ans+15
```

Tout d'abord on a déclaré la List Ans en lui attribuant les valeurs 5,2,3,5 dans les cases 1 à 4 :

List Ans à la ligne 1

Case	Valeur
1	5
2	2
3	3
4	5

La dimension de la List Ans est désormais égale à 4. Si la List Ans existait déjà, elle a été supprimé avant qu'on la crée, et si elle n'existait pas, ça l'a crée.
A la deuxième ligne, on ajoute 15 à toutes les cases de la List Ans, qui se présentera désormais comme ceci :

List Ans à la ligne 2

Case	Valeur
1	20
2	17
3	18
4	20

Au lieu de travailler sur la List Ans, on aurait pu tout aussi bien le faire sur la List 1 :

Code : Autre - Addition d'une liste et d'un nombre

```
{5,2,3,5}->List 1
List 1+15->List 1
```

On peut aussi faire la somme de plusieurs listes, **à condition qu'elles aient la même dimension**, sinon vous aurez un **Dimension ERROR**. Vous allez comprendre pourquoi : les opérations entre listes se font case par case.

Code : Autre - Addition de listes

```
{1,2,3,4}->List 1
{5,2,10,5}->List 2
{5,3,1,9}+List 1+List 2
```

Comme le résultat de la dernière ligne n'est pas attribué à une liste, il est attribué à List Ans, qui sera remplie de la façon suivante :

List Ans à la dernière ligne

Case	Valeur
1	11
2	7
3	14
4	18

Quelques généralités :

Les crochets permettent d'accéder à une case d'une liste.
On ne peut pas accéder à une case d'une liste dont le numéro est supérieur à la dimension de cette liste.
On peut cependant attribuer une valeur à la case N+1 d'une liste de dimension N, ce qui fait que la liste s'agrandit d'une case (**uniquement sur Graph 85 !**).
Tout cela n'est possible que pour une liste existante (de dimension supérieure ou égale à 1), si ce n'est le cas vous aurez un **Dimension ERROR**.

La dimension maximale d'une liste est de 999 sur Graph 85, et 255 sur les autres Graph.

LES FONCTIONS SPÉCIFIQUES

Dans cette sous-partie, vous allez peut-être vous ennuyer. Je ne vous en voudrais pas (des fois que vous culpabilisiez).
En fait, cette sous-partie est plutôt une documentation non-officielle. Lisez quand même le descriptif de Dim, qui est essentiel pour utiliser les listes.

Dim liste

=> [OPTION][F1][F3]
Voici une fonction que vous devez à tout prix savoir utiliser !
Elle peut s'utiliser de deux manières différentes :

- elle renvoie un entier naturel non nul, qui n'est autre que la dimension de *liste*. Si la liste n'existe pas, vous aurez un **Dimension ERROR**.

Code : Autre - Exemple d'utilisation de Dim pour récupérer la dimension d'une liste

```
{5,32}
Dim List Ans->A
```

A la dernière ligne, A vaut 2.

- elle crée une liste de la dimension égale au nombre qu'on lui attribue (entier naturel non nul obligatoire). Cette fonction ne fonctionne pas sur la Graph 35 (ancêtre de la Graph 35+).

Code : Autre - Exemple d'utilisation de Dim pour créer une liste

```
5->Dim List 1
```

Sachez qu'on ne peut pas faire ça pour la List Ans. Pour les autres listes, en plus d'être créées, leurs cases seront initialisées à 0.

Fill(valeur,liste)

=> [OPTION][F1][F4]
Cette fonction remplit la liste *liste* avec *valeur*.

Code : Autre - Exemple d'utilisation de Fill(

```
5->Dim List 1
Fill(10.5,List 1)
```

List 1 à la dernière ligne

Case	Valeur
1	10.5
2	10.5
3	10.5
4	10.5
5	10.5

Seq(expression,variable,valeur initiale,dimension,pas)

=> [OPTION][F1][F5]
Cette fonction renvoie une liste composée de *dimension* cases dont toutes les valeurs respecteront l'*expression*, avec *variable* égale à *valeur initiale*, à laquelle est ajoutée *pas* à chaque case.

Code : Autre - Exemple d'utilisation de Seq(

```
Seq(X^2,X,0,4,1)
```

List Ans

Case	Valeur
1	0
2	1
3	4
4	9
5	16

Augment(première liste, deuxième liste)

=> [OPTION][F1][F6][F5]
Cette fonction renvoie une liste, qui est le résultat de la *première liste* et *deuxième liste* mises bout à bout (ça pourrait rappeler à certains la concaténation, très utilisée en PHP, par exemple).

Code : Autre - Exemple d'utilisation d'Augment(

```
{15,10,8,8,7}->List 1
{6,5,10}
Augment(List Ans,List 1)
```

List Ans à la dernière ligne

Case	Valeur
1	6
2	5
3	10
4	15
5	10
6	8
7	8
8	7

Cuml liste

=> [OPTION][F1][F6][F6][F3]
Cette fonction renvoie une liste, qui est l'ensemble des effectifs cumulés.

Code : Autre - Exemple d'utilisation de Cuml

```
Cuml {15,10,8,8,7}
```

List Ans

Case	Valeur
1	15
2	25
3	33
4	41
5	48

Percent liste

=> [OPTION][F1][F6][F6][F4]
Cette fonction renvoie une liste : les pourcentages de la liste *liste*.

Code : Autre - Exemple d'utilisation de Percent

```
Percent {5,13,12,8,2}
```

List Ans

Case	Valeur
1	12.5
2	32.5
3	30
4	5
5	20

ΔList liste

=> [OPTION][F1][F6][F6][F5]
Cette fonction renvoie une liste de dimension (Dim *liste*-1) contenant les variations de la liste *liste*.

Code : Autre - Exemple d'utilisation de ΔList

```
△List {5,13,12,8,2}
```

List Ans

Case	Valeur

Case	Valeur
1	-8
2	-1
3	-10
4	6

Sum(liste)

=> [OPTION][F1][F6][F6][F1]
Cette fonction renvoie la somme des cases de la liste *liste*.

Code : Autre - Exemple d'utilisation de Sum

```
Sum {15,10,8,8,7}
```

Ans vaut désormais $\sum_{I=0}^{N}\{15,10,8,8,7\}[I]=15+10+8+8+7=48$ (avec N la dimension de la liste : 5).

Prod(liste)

=> [OPTION][F1][F6][F6][F2]
Cette fonction renvoie le produit des cases de la liste *liste*.

Code : Autre - Exemple d'utilisation de Prod

```
Prod {15,10,8,8,7}
```

Ans vaut désormais $\prod_{I=0}^{N}\{15,10,8,8,7\}[I]=15*10*8*8*7=67200$ (avec N la dimension de la liste : 5).

Min(liste)

=> [OPTION][F1][F6][F1]
Cette fonction renvoie la plus petite valeur de la liste *liste*.

Code : Autre - Exemple d'utilisation de Min(

```
Min({15,10,8,8,7})
```

Ans vaut désormais 7.

Max(liste)

=> [OPTION][F1][F6][F2]
Dans le même genre que Min(, cette fonction renvoie la plus grande valeur de la liste *liste*.

Code : Autre - Exemple d'utilisation de Max(

```
Max({15,10,8,8,7})
```

Ans vaut désormais 15.

Mean(liste)

=> [OPTION][F1][F6][F3]
Cette fonction renvoie la valeur moyenne de la liste.

Code : Autre - Exemple d'utilisation de Mean(

```
Mean({15,10,8,8,7})
```

Ans vaut désormais 9.6.

Median(liste)

=> [OPTION][F1][F6][F3]
Cette fonction renvoie la **valeur médiane** de la liste (si la liste est de dimension paire, c'est la valeur de la case du milieu, sinon, c'est la moyenne entre les deux cases du milieu).

Code : Autre - Exemple d'utilisation de Median(

```
Median({15,10,8,8,7})
```

Ans vaut désormais 8.

List->Mat(liste)

=> [OPTION][F1][F2]
Cette fonction est un peu différente des autres, elle transforme la liste *liste* en matrice à une dimension.

Code : Autre - Exemple d'utilisation de List->Mat(

```
List->Mat({15,22,30})->Mat A
```

Mat->List(matrice,colonne)

=> [OPTION][F2][F1]
Cette fonction ressemble à la précédente, sauf qu'elle fait l'opération inverse : elle transforme la colonne *colonne* de matrice *matrice* en liste.

Code : Autre - Exemple d'utilisation de List->Mat(

```
Mat->List([[15,2,54][24,3,8]],3)
```

Ca faisait longtemps qu'on avait pas vu un petit tableau, non ?

List Ans à la dernière ligne

Case	Valeur
1	54
2	8

SortA(liste)

=> [F4][F3][F1]
Cette fonction modifie la liste *liste* en la triant par ordre croissant. *liste* est forcément une liste numérotée (pas de List Ans), sinon vous aurez un **Argument ERROR**.

Code : Autre - Exemple d'utilisation de SortA(

```
{12,51,9,23}->List 1
SortA(List 1)
```

List 1 à la dernière ligne

Case	Valeur
1	9
2	12
3	23
4	51

SortD(liste)

=> [F4][F3][F2]
Cette fonction fonctionne comme SortA(, sauf qu'elle trie par ordre décroissant.

File numéro

=> [SHIFT][MENU][F6][F6][F1]
Cette fonction ouvre le File *numéro*. Il faut savoir qu'il y a 6 File (littéralement "fichier"), contenant chacun un paquet de listes (26 pour la Graph 85, 6 pour les autres). Au cours d'un programme, vous pouvez changez de File, ce qui vous permettra de stocker plus de listes et ce de manière plus durable. La List Ans reste commune à tous les File, ce qui permet de faire des transferts entre File.

Code : Autre - Exemple d'utilisation de File

```
File 3
{12,65,21}->List 1
File 2
{0,32,84,1}->List 1
File 3
```

List 1 à la dernière ligne

Case	Valeur
1	12
2	65
3	21

ClrList numéro

=> [SHIFT][VARS][F6][F1][F3]

Cette fonction supprime la liste de numéro *numéro*. *numéro* peut être Ans (auquel cas la List Ans est supprimée). Vous pouvez aussi ne pas mettre de *numéro*, cela supprimera alors toutes les listes.

La plupart des fonctions qu'on a vu dans cette sous-partie sont des fonctions statistiques, qui peuvent être bien utiles pour le calcul. Désormais, on va apprendre à dessiner avec les listes, et croyez-moi, vous allez arrêter d'utiliser les F-Line dans peu de temps .

DRAWSTAT, LE DESSIN-ÉCLAIR

Le DrawStat est la fonction graphique la plus intéressante d'après moi. Elle permet de tracer des points, des lignes (des boîtes à moustaches, aussi, mais ça m'étonnerait que vous vous en serviez dans votre programme) à une vitesse surprenante.

Jusqu'à maintenant, comment traciez-vous vos lignes ?

- avec Line, qui relie entre eux les deux derniers Plot lus dans le programme. C'est la pire de toutes les méthodes : elle prend trois lignes (deux Plot + 1 Line) et est très, très lente.
- avec F-Line, qui relie deux points entre eux, avec une vitesse acceptable, et ne prenant qu'une seule ligne.

Mais il y a mieux que tout ça : le DrawStat. (Très) Rapide, compact, adapté aux transformations géométriques du plan (et de l'espace, mais c'est plus complexe). Le DrawStat prend en paramètre une liste pour les abscisses des points à tracer, et une liste pour leurs ordonnées.
Une translation ? Rien de plus facile : une seule addition vous permet de translater tout un dessin, selon l'axe des abscisses ou des ordonnées.
Une rotation ? Pareil, on peut utiliser les fonctions trigonométriques pour faire tourner une figure autour d'un point.
Symétrie axiale ? Du gâteau...

Néanmoins, le DrawStat n'est pas parfait. Premièrement, il faut faire pas mal de réglages au début du programme. Avant, on se contentait de faire

Code : Autre - Anciens réglages

```
ViewWindow 1,127,0,1,63,0
```

Quand on utilise le DrawStat, les réglages sont plus longs. Il faut indiquer quels types de dessin il faut faire, et ce pour les 3 S-Gph (littéralement "graphiques statistiques"), c'est-à-dire si l'on veut des lignes, des points, une boîte à moustache ~~ou une perruque~~, si l'on veut telle ou telle liste pour coordonnées, ...

Ici, on ne va pas aborder l'aspect statistique du DrawStat, on ne va parler que des points et des lignes.
Cela me paraît justifié du fait que je n'ai jamais vu personne avoir envie de faire une boîte à moustache dans un programme. Si vous pensez le contraire, faites-m'en part dans les commentaires, vous pourriez alors me faire changer d'avis

Pour dire au DrawStat qu'il faut suivre la ViewWindow comme S-Window ("Stat-Window"), il faut utiliser S-WindMan. Personnellement, j'aime bien fixer ma ViewWindow au début de mes programmes, et je mets S-WindMan juste avant (la Stat-Window se met à jour à chaque changement de ViewWindow) ou juste après.

Le plus dur reste à venir (je blague) : il faut définir ce qu'il faut dessiner et comment le dessiner.
Le DrawStat peut faire trois tracés différents, que l'on appelle S-Gph, numérotés de 1 à 3.
On définira les modes de tracé pour chacun de ces 3 Graphs comme ceci : **S-GphN *état*,*type*,*abscisses*,*ordonnées*,*1*,*motif***, avec :

- S-GphN le S-Gph à définir.
  [F4][F1][F2][F1~F3]
- *état* valant soit DrawOn, soit DrawOff, c'est-à-dire soit activé, soit désactivé (et oui, on a pas forcément besoin des 3 S-Gph).
  [F4][F1][F1][F1~F2]
- *type* valant soit xyLine, soit Scatter, c'est-à-dire soit "ligne" (les points définis par les listes sont reliés entre eux), soit "point" (comme des Plot).
  [F4][F1][F2][F4~F5]
- *motif* valant soit Dot (pixel), soit Cross (croix), Square (carré). On utilisera le plus souvent Dot.
  [F4][F1][F4][F1~F3]

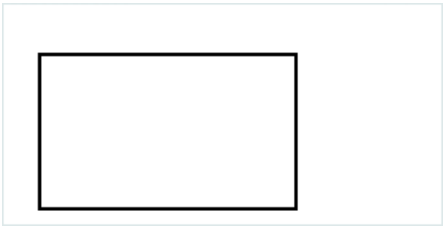
Quand vous appellerez DrawStat, cela tracera tous les S-Gphs activés.
Pour les S-Gphs que vous n'allez pas du tout utiliser dans votre programme, il ne sert à rien de définir leur mode de tracé, à moins que vous ne les désactiviez que temporairement.

On va commencer à pratiquer : commençons par dessiner un rectangle.

Code : Autre - Accrochez vos ceintures !

```
ViewWindow 1,127,0,1,63,0
S-WindMan
S-Gph2 DrawOff
S-Gph3 DrawOff
S-Gph1 DrawOn,xyLine,List 1,List 2,1,Dot
'n'oubliez pas de faire relier le premier et le dernier point
{10,10,85,85,10}->List 1
{5,50,50,5,5}->List 2
DrawStat
```

Vous avez fait votre premier dessin en DrawStat :



Je ne sais pas si vous avez remarqué lors du tracé que "StatGraph1" s'est écrit en haut à gauche. Pour remédier à ce problème parasite, il faut appeler la fonction FuncOff. Généralement, je la mets à côté de S-WindMan, et je n'y touche plus.

Reprenons nos exercices pratiques, en essayant de tracer un rectangle et un triangle.

Un problème se pose : comment éviter de relier les sommets des deux polygones ?

Certains diront qu'il faut faire deux DrawStat, ou utiliser deux S-Gph. Le problème avec ces deux solutions, c'est que c'est lourd, lent et pas pratique. Le DrawStat est rapide, mais il vaut mieux éviter de lui faire faire plein de petits dessins, il préfère les gros.

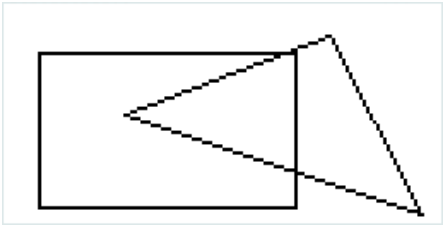
La solution consiste en l'utilisation d'une séparation : il faut créer un point dont les coordonnées seront entre celles des deux polygones, et dont la valeur dépassera le cadre de la ViewWindow. Dans ce cas, il n'y aura ni trait reliant le dernier sommet du premier polygone et le point séparateur, ni trait reliant ce point au premier sommet du dernier polygone.
Dans notre cas, la ViewWindow est limitée par 127 en abscisses et 63 en ordonnées. Il faudra donc, à chaque séparation, tapez une valeur strictement supérieure à 127 dans la liste des abscisses et une valeur strictement supérieure à 63 dans la liste des ordonnées.

C'est assez fastidieux, il faut l'avouer, et c'est pour ça que je vous conseille très fortement d'utiliser une variable qui ne servira qu'à contenir cette valeur, et que vous initialiserez en début de programme.

Code : Autre - Rectangle et triangle en un seul DrawStat

```
ViewWindow 1,127,0,1,63,0
S-WindMan
FuncOff
128->D
S-Gph2 DrawOff
S-Gph3 DrawOff
S-Gph1 DrawOn,xyLine,List 1,List 2,1,Dot
{10,10,85,85,10,D,122,95,35,122}->List 1
{5,50,50,5,5,D,3,55,32,3}->List 2
DrawStat
```

Sans surprise, on voit s'afficher :



Scatter fonctionne comme xyLine, sauf qu'il ne relie pas les points. Je pense que vous n'aurez pas de difficultés à l'utiliser .

Vous rappelez-vous de ce qu'on a vu dans les précédents chapitres ?

Voici quelques exemples que vous pourrez vous amuser à utiliser :

- L'addition sur les listes vous permet de faire des translations (faites attention à ce que la séparation reste toujours en dehors de la ViewWindow)
- La multiplication vous permet de faire des homothéties
- Seq(vous aidera à faire des pointillés (en mode Scatter)

On va faire un troisième exercice : dessiner un rectangle, mettre une pause dans le programme (tant qu'on a pas appuyé sur [F1]), puis dessiner, en un seul DrawStat, une ligne pointillée verticale et le rectangle précédent translaté de 5 pixels vers la droite.

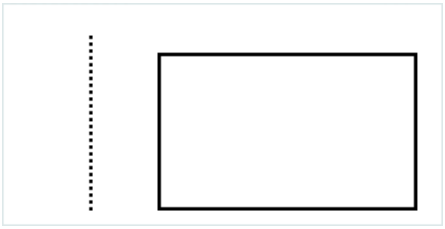
Comme je suis un gentil diable, je vous donne une solution :

Code : Autre - Solution d'un exo sadique

```
ViewWindow 1,127,0,1,63,0
S-WindMan
FuncOff
128->D
S-Gph2 DrawOff,Scatter,List 3,List 4,1,Dot
S-Gph3 DrawOff
S-Gph1 DrawOn,xyLine,List 1,List 2,1,Dot
{10,10,85,85,10}->List 1
{5,50,50,5,5}->List 2
DrawStat
Do
LpWhile Getkey!=79
'et oui, on peut juste faire DrawOn (vu qu'on l'a déjà configuré, il n'y a pas de problème)
S-Gph2 DrawOn
List 1+35->List 1
Seq(X,X,5,55,2)->List 3
Dim List 3->Dim List 4
Fill(25,List 4)
DrawStat
```

On aura donc en premier le rectangle de tout à l'heure :

Et une fois qu'on a appuyé sur [F1] :



Un conseil : n'utilisez pas _Disp_ (le triangle noir qui crée une pause tant qu'on ne presse pas [EXE]), car il suffirait qu'on appuie sur une des flèches pour changer la ViewWindow et effacer l'écran (seul le dernier DrawStat restera affiché, en décalé du fait du changement de la ViewWindow).

Liste des fonctions utilisées et leur chemin d'accès

- DrawStat : [SHIFT][VARS][F6][F2][F1]
- S-WindMan : [SHIFT][MENU][F6][F6][F3][F2]
- FuncOff : [SHIFT][MENU][F6][F6][F1][F2]
- S-Gph1 : [F4][F1][F2][F1] (pour S-Gph2 et S-Gph3, je vous laisse chercher)
- Scatter : [F4][F1][F2][F4]
- xyLine : [F4][F1][F2][F5]
- DrawOn : [F4][F1][F1][F1]
- DrawOff : [F4][F1][F1][F2]
- Square : [F4][F1][F4][F1]
- Cross : [F4][F1][F4][F2]
- Dot : [F4][F1][F4][F3]

Cette dernière sous-partie est terminée, j'espère qu'elle vous aura donné plein d'idées de graphismes ambitieux, que seul le DrawStat peut afficher avec fluidité. Faites-nous de beaux programmes rapides

---

Si ce tuto vous a plu, sachez qu'il y en aura sûrement d'autre (surtout si vous m'écrivez plein d'avis encourageants).
Je m'excuse encore d'avoir fait une seconde partie aussi barbante, je n'ai pas le don d'amuser la galerie en "enseignant" (je m'appelle pas M@teo21).
Ce tutoriel n'est pas figé, il y aura peut-être des TP dans une future mise à jour, voire un big-tuto sur le Basic Casio, dans son ensemble.

Merci de m'avoir lu , passez faire un p'tit coucou dans les commentaires.

Remerciements à Planete-Casio et à ses membres, en particulier **PierrotLL** pour ses lectures, relectures et avis avisés, ainsi que **Thomatos** (qui m'a fourni l'icône que j'ai modifié pour le rendre carré).

Auteur : pylaterreur