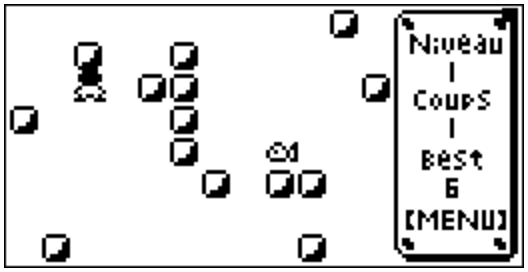
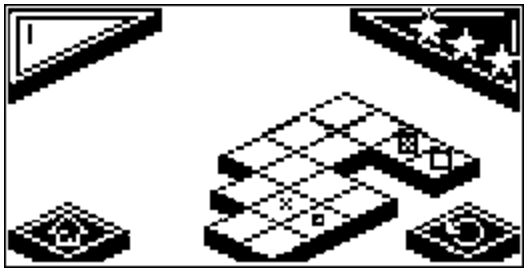
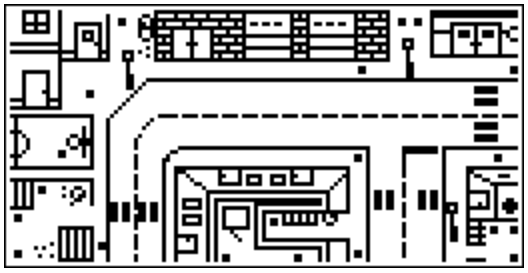
Débuter la programmation sur calculatrice

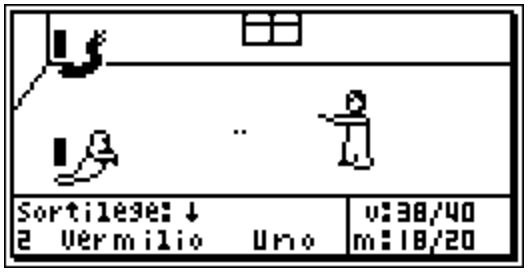
Bonjour à tous ! Nous allons commencer par nous présenter : nous sommes trois programmeurs actifs de Planète-Casio (communauté d'échange autour des calculatrices) : Louis GATIN (Dark storm), Paul GODEFROY (Purobaz) et Louis SUGY (Louloux). Notre objectif est de vous apprendre à faire des jeux sur votre calculatrice. Si vous suivez ce cours et que vous continuez à vous intéresser à la programmation, vous serez bientôt en mesure de faire des trucs vraiment sympas, comme ces jeux réalisés par la communauté :



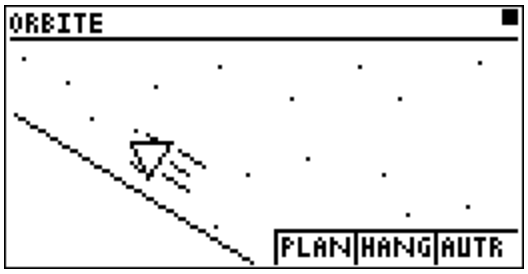
Ice Slider (Ne0tux)


CloneLab (Ne0tux)

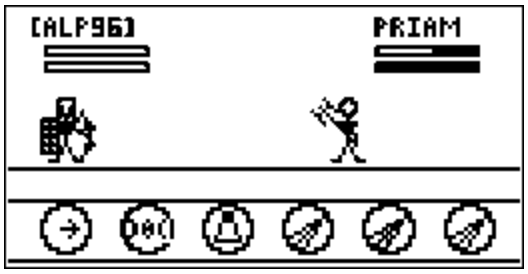

Drug Life (Neuronix)



Harry Potter (Samy)



Pirate Galaxy (Louloux)



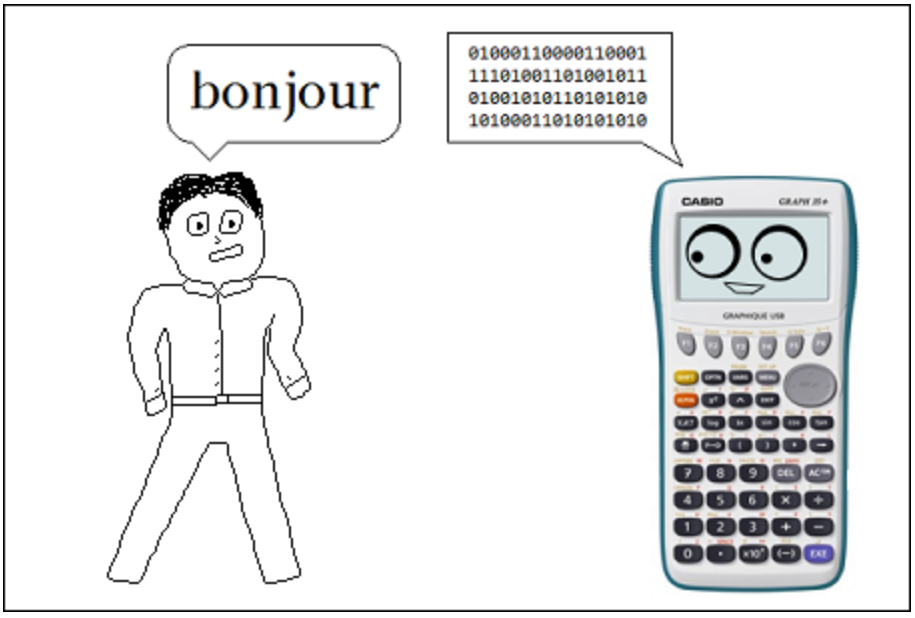
Sword Sandals (Alp96)

Je sens déjà monter dans vos esprits la lourde question : « Mais comment ont-ils fait ça ?? ». Alors nous allons vous faire un petit exposé sur la programmation, puis dans ce cours nous en verrons les bases. Dans les cours suivants vous verrez beaucoup de notions très utiles pour démarrer la programmation, nous allons apprendre tout ça pas à pas. Prêts ? Faites chauffer vos calculatrices, on démarre !

Cours de programmation réalisé par Louis SUGY pour la page Facebook Casio Calculatrices.

C’est quoi la programmation ?

Pour faire simple, c’est dialoguer avec votre machine. Sauf que vous vous rendez vite compte que votre calculatrice et vous ne parlez pas tout à fait de la même façon :



Au début on a dû se plier à la simplicité du langage de nos machines, et on faisait des calculs avec des cartes perforées, donc évidemment ce n’était accessible qu’aux scientifiques… Mais peu à peu les génies de l’informatique ont conçu ces machines de telle manière qu’on puisse les programmer beaucoup plus simplement, et de nombreux langages de programmation sont apparus, tels que maintenant tout le monde (ou presque) est capable de s’en servir.

L’entité de base du programme est l’algorithme. Pour faire simple, c’est une suite organisée d’instructions qui à partir d’entrées rend des sorties. Prenons par exemple l’algorithme du carré :

- on entre 2, il sort 4
- on entre 5, il sort 25
- on entre 9, il sort 81
  etc…

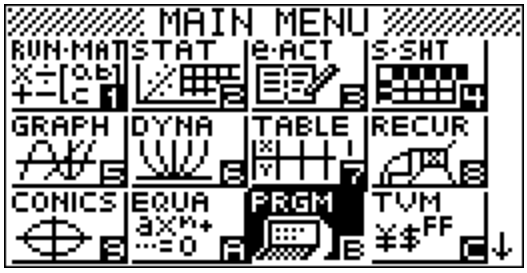
Bien sûr l’algorithme peut faire des choses bien plus complexes que de petits calculs mathématiques. Je pense notamment aux prévisions météo, à la localisation GPS, à la desserte du réseau par les opérateurs téléphoniques, ou aux zombies qui vous affrontent sur votre PS3.

Et nous-mêmes allons faire pas mal d’algorithmes tout au long de ce cours.

Bon, assez discuté, nous allons maintenant découvrir le menu *Programmes* de notre calculatrice.

Le menu *Programmes* des Casio

Rendez-vous dans le menu *Programmes*, reconnaissable par l'icône **PRGM** avec un ordinateur, comme indiqué ci-dessous :



Si il n'est pas situé au même endroit dans votre menu, c'est parfaitement normal, rassurez-vous : le modèle montré ici possède des applications supplémentaires que la majorité d'entre vous n'a pas (*e-activity* et *s-sheet*, en haut à droite).

Maintenant nous allons découvrir ce menu. Si vous n'avez encore aucun programme, vous obtiendrez l'image de gauche. Si vous avez des programmes ça ressemblera plutôt à l'image de droite, avec la liste des programmes (désolé de casser vos espoirs de gamers, ceux-ci sont fictifs) :

Il y a pas mal d'options ici. C'est de l'anglais, mais comme je suis sympa je vais traduire et interpréter ces commandes : [F1] (EXE) exécute le programme sélectionné, [F2] (EDIT) permet de l'éditer, [F3] (NEW) d'en créer un nouveau, [F4] (DEL) de supprimer le programme sélectionné, [F5] (DEL-A) de les supprimer tous, [F6] d'afficher les autres commandes. Lorsque les autres commandes sont affichées : [F1] (SRC) permet de rechercher un programme dans la liste et [F2] (REN) de renommer le programme sélectionné.

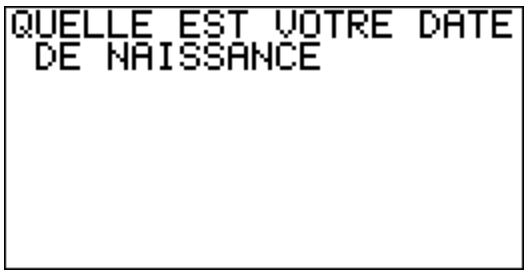
Bon, maintenant créons un programme, et nous allons pouvoir étudier ces menus. Nous l'appellerons « PROG1 ». Pour créer un programme, il faut faire [F3] (NEW), puis entrer le nom du programme à l'aide du clavier, et enfin appuyer sur [EXE]. Vous arrivez ensuite dans l'écran de saisie des instructions :

A partir d'ici nous pouvons commencer à voir les fondements du Basic Casio, le langage de programmation dans avec vous allez faire vos premiers programmes.

Pour apprendre les premières notions nous allons réaliser un programme qui gère une boutique de vente d'alcool. Il va demander à l'utilisateur sa date de naissance, calculer l'âge de l'utilisateur, et dire s'il a le droit ou non d'acheter de l'alcool.

Afficher du texte

La première chose que nous avons à faire est donc d'afficher du texte. Pour cela nous allons simplement écrire entre guillemets le texte dans le programme, puis on va sortir et exécuter :



Notre texte s'est affiché, et en plus il est revenu à la ligne tout seul. On se rend compte qu'on peut supprimer l'espace entre « DATE » et « DE », pour faire plus joli.

Maintenant qu'on a posé la question à l'utilisateur, il faudrait qu'il puisse y répondre. Nous allons avoir besoin de variables pour cela.

Les variables

Ne vous affolez pas, même si vous venez de rencontrer le premier terme technique. Vous verrez que c'est la notion la plus utile en programmation et qu'on ne peut rien faire sans.

En fait, la variable est une valeur stockée en mémoire pendant l'utilisation d'un programme, et que l'on peut changer, sur laquelle on peut faire des calculs, etc. En Basic on la note avec une lettre majuscule qui, contrairement à un texte, n'est pas mise entre guillemets. Par exemple, si on veut utiliser la variable A pour stocker l'âge de l'utilisateur, dès qu'on lui aura donné une valeur elle la gardera jusqu'à ce qu'on change cette valeur.

Pour définir une variable on écrit :

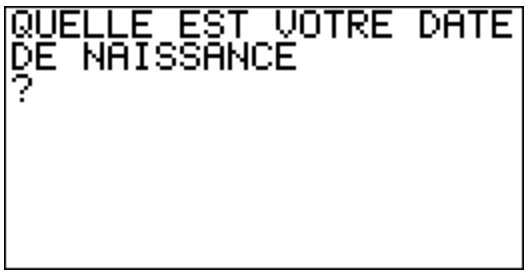
Valeur → *variable* (la flèche est sur votre clavier, au-dessus de la touche [AC/ON])

Par exemple, on peut donner la valeur 16 à A en faisant : 16→A.
Et ensuite on peut faire des calculs avec cette variable, comme : A+3→B (B vaudra 19).
Et on peut la modifier sans limite, en faisant par exemple : Ax2→A (A vaudra maintenant 32).

Mais comment demander à l'utilisateur de donner une valeur à sa date de naissance ? Notons D la variable qui prendra pour valeur la date de naissance. Nous allons utiliser l'instruction **?**. Oui, vous avez bien lu : le point d'interrogation est une instruction en Basic. Pour le trouver faites [SHIFT] / [VARS] / [F4]. Voilà comment il faut l'utiliser :

? → *variable*

Nous allons donc compléter notre programme. Entre deux instructions il faut ajouter une flèche de changement d'instruction en appuyant sur [EXE]. Voilà ci-dessous ce que vous devez avoir. A gauche le code et à droite l'exécution :



Et là vous pouvez déjà écrire en nombre, appuyer sur [EXE], et il est stocké dans D. Mais après le programme ne fait pas grand-chose. Alors on va déjà commencer par le calcul de l'âge. Comme on l'a vu juste avant, nous pouvons aisément faire un calcul sur la variable D. L'âge correspond à l'année actuelle (2013) moins l'année de naissance (c'est certes très approximatif, car on ne connaît pas le mois et le jour de naissance). Allez, on complète l'algorithme :

Ok, on connaît l'âge de l'utilisateur, mais il faut maintenant trouver un moyen de lui dire si on peut le servir ou pas. Nous allons donc découvrir ensemble les conditions.

Les conditions

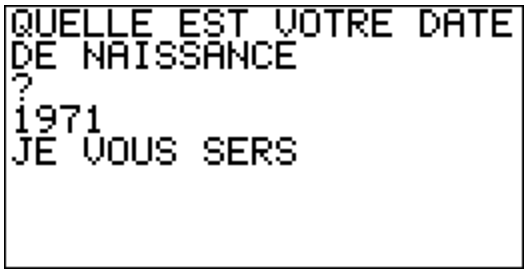
Encore une notion importante de l'algorithmique que voilà ! Elle permet tout simplement de tester une propriété et de choisir entre deux issues : si la condition est respectée, on fait ça, et sinon on fait ça. En Basic la syntaxe est évidemment tirée de l'anglais, et ça donne :

If *condition*
Then *instructions*
Else *instructions*
IfEnd

Toutes les fonctions conditionnelle se trouvent dans le menu [SHIFT] / [VARS] / [F1] (COM).

Et maintenant, complétons notre algorithme sans perdre de temps. Pour la condition vous trouverez les comparateurs dans [SHIFT] / [VARS] / [F6] / [F3] (REL). Nous allons afficher « PAS LE DROIT » si l'utilisateur est mineur, et « JE VOUS SERS » s'il est majeur. Vous devriez avoir ce qui suit :

Testons avec les dates 1971 et 1998 :



Chouette, ça a marché !

Travaillons maintenant sur un nouveau problème : votre calculatrice gère l'ascenseur du lycée. Mais pour éviter que celui-ci ne soit trop emprunté par les élèves, les professeurs ont fait un complot terrible : ils ont mis un code. Votre calculatrice va donc attendre que vous saisissiez le bon code pour ouvrir l'ascenseur. Comment on va faire ça ? Nous allons vous présenter les boucles conditionnelles. Créez un nouveau programme, et on attaque !

Les boucles conditionnelles

Quand on vous dit le mot « boucle », vous pensez immédiatement à un truc qui tourne et revient à son point de départ, non ? Eh bien c'est exactement pareil en algorithmique ! La boucle conditionnelle tourne tant qu'une condition est remplie, et si la condition devient fausse on en sort.

La boucle conditionnelle existe sous deux formes, selon que la condition est placée au début ou à la fin. En français on dit « tant que… », donc il est assez logique que l'équivalent anglais soit « while… ». La différence entre les deux formes est que si la condition est à la fin la boucle fait au moins un tour, alors que si la condition est au début et qu'elle n'est pas remplie dès le départ, la boucle n'est pas exécutée du tout.

Voilà la syntaxe des deux :

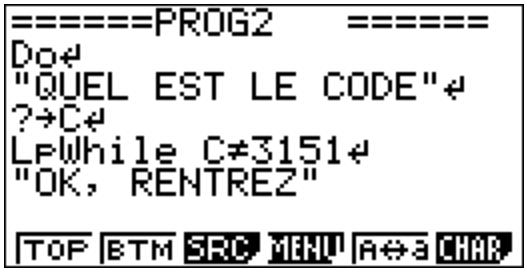
- **While** *condition*
  Instructions
  WhileEnd

- Do
  Instructions
  LpWhile *condition*

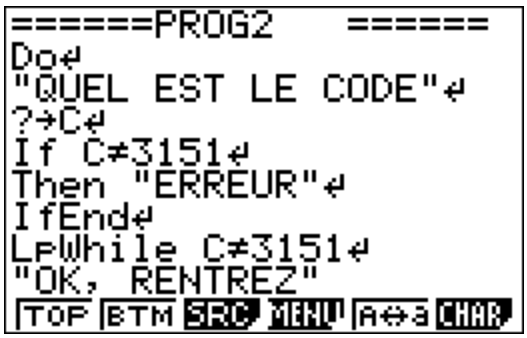
Vous pouvez trouver ces commandes dans le même sous-menu que les instructions conditionnelles : [SHIFT] / [VARS] / [F1] (COM), en faisant défiler les instructions de ce sous-menu avec [F6].

On va dire que l'ascenseur de notre exemple a pour code 3151. La boucle va donc tourner tant que l'utilisateur n'aura pas rentré le bon code. Nous allons stocker le code rentré par l'utilisateur dans la variable C, et nous servir de la seconde boucle conditionnelle présentée.

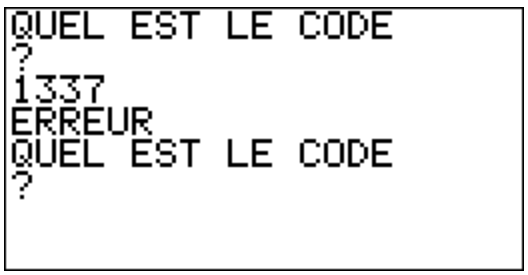
Réfléchissez un instant, puis regardez la solution que nous proposons :

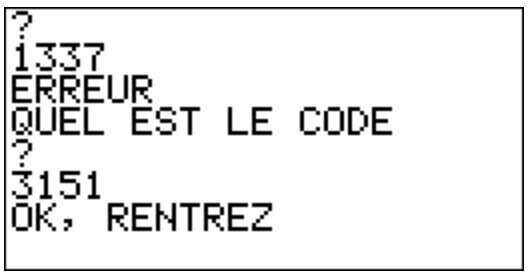


En ajoutant une condition à l'intérieur de la boucle, on peut afficher « ERREUR » lorsque l'utilisateur rentre un faux code. On essaie ça ? Voilà la soluce :



Essayons avec un mauvais code, puis le bon :





Maintenant, l'utilisateur est dans l'ascenseur et il monte. C'est un ascenseur qui demande à quel étage on veut aller, et à chaque étage s'arrête, affiche le numéro de l'étage, attend qu'on appuie sur un bouton et redémarre. Comment peut-on faire ça ? On pourrait toujours imaginer une boucle conditionnelle, mais on peut faire bien plus simple : la boucle d'incrémentation, ou « boucle pour ». Créez un nouveau programme, et on y va !

La boucle pour

Un super système, cette boucle ! La variable d'incrémentation définie prend une valeur initiale, augmente ou diminue à chaque tour selon le pas réglé, et la boucle tourne tant qu'elle n'a pas atteint sa valeur finale. Voilà sa syntaxe :

For *valeur initiale* → *variable* **To** *valeur finale* **Step** *pas*
Instructions
Next

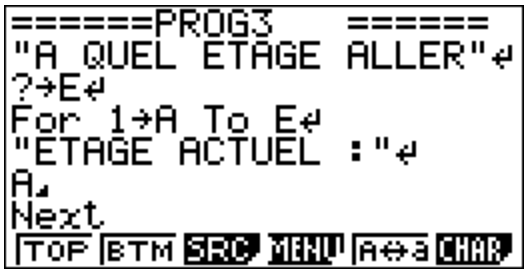
Si vous ne réglez pas de pas (vous n'écrivez pas l'instruction **Step**) il est automatiquement de 1.

Il existe deux cas de figure :

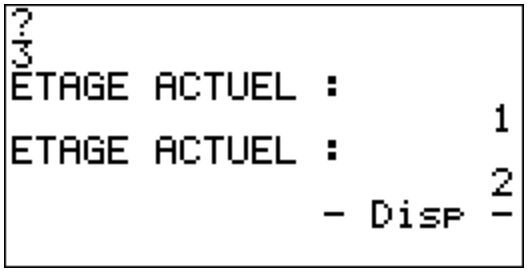
- Valeur finale plus grande que la valeur initiale = le pas est positif.
- Valeur initiale plus grande que la valeur finale = le pas est négatif.

Attaquons notre programme de l'ascenseur ! Nous aurons besoin d'une fonction très pratique, le petit triangle accessible avec [SHIFT] / [VARS] / [F5]. Il permet de mettre en pause le programme jusqu'à pression de [EXE] par l'utilisateur, et, lorsqu'il est écrit juste à côté d'un nombre, de l'afficher. Il fait office de changement d'instruction, à la place de la flèche de fin de ligne.

Notons E l'étage où l'utilisateur veut monter et A l'étage actuel. Nous avons donc :



Testons avec l'étage 3 :



Et maintenant nous pouvons compléter l'algorithme pour empêcher de saisir un nombre plus grand que le nombre d'étages (il y en a 5), et afficher « VOUS ETES ARRIVE » à la fin.

Comme les algorithmes que nous allons faire dorénavant sont un peu plus longs, nous n'allons plus vous les montrer en capture d'écran mais dans des zones comme la suivante. Nous afficherons les commentaires sur les instructions en vert après un double-slash. Voilà le code :

```
"A QUEL ETAGE ALLER"
?→E
If E>5
Then "IL Y A QUE 5 ETAGES"
5→E // si l'utilisateur dépasse le nombre d'étages on l'emmène au plus haut
IfEnd
For 1→A To E
 "ETAGE ACTUEL : "
A◢
Next
"VOUS ETES ARRIVE"
```

Normalement votre ascenseur devrait vous conduire avec succès. Bien sûr il faudrait complexifier un peu si on ne partait pas du rez-de-chaussée, mais nous pouvons déjà être satisfaits de ces premiers résultats.

Ce cours est terminé, vous avez pu voir l'affichage de texte, les variables, les conditions et les boucles.

Le prochain cours concernera les fonctions les plus usuelles du langage de programmation de votre calculatrice. A bientôt !