Journal : TapTempo en emacs lisp

Posté par **omc** le 12/03/18 à 11:36. **Licence CC by-sa**
Tags : **taptempo**, **elisp**

Le portage de TapTempo en emacs lisp est idéal pour se muscler les doigts !
Battez la mesure sous votre éditeur préféré à coup `C-c C-s` et vous aurez votre bpm dans le mini-buffer.

À rajouter d'urgence dans votre `.emacs`

```
(setq t0 0)
(setq array (list))

(defun sum(list)
  (if (null list)
      0
    (+
     (first list)
     (sum (rest list))
     )
    )
  )


(defun tap ()
  "Record a tap"
  (interactive)
  (setq t1 (float-time()))
  (setq delta (- t1 t0))
  (setq freq (/ 1 delta))
  (setq bpm (* 60 freq))
  (push bpm array)
  (setq res (/ (sum array) (length array)))
  (message "BPM %S" res)
  (setq t0 t1))

(global-set-key "\C-x\C-s" 'tap)
```

Oups

Posté par **Christie Poutrelle** (**page perso**) le 12/03/18 à 12:05. Évalué à 3 (+3/-1).

> *Battez la mesure sous votre éditeur préféré*

Dans mon Eclipse, ça ne marche pas. Tu mens!

Re: Oups

Posté par **omc** le 12/03/18 à 12:09. Évalué à 10 (+14/-0).

Emacs est bien ton éditeur préféré mais tu ne le sais pas encore !

Re: Oups

Posté par **nigaiden** le 12/03/18 à 12:45. Évalué à 10 (+10/-0).

C'est mal parti : le programme proposé bloque les sauvegardes des autres fichiers et en plus il ne s'exécute pas ("first" est un symbole not found visiblement).
Heureusement Vim était là pour me dépanner.

Re: Oups

Posté par **omc** le 12/03/18 à 19:35. Évalué à 3 (+4/-2).

J'ai beau faire `M-x vim` dans mon mini-buffer, cela ne donne rien `[No match]`. Qu'est-ce donc ?

Bouh !

Posté par **Axioplase ıɥsɐ∀** (**page perso**) le 12/03/18 à 14:04. Évalué à 4 (+2/-0).

D'abord, la récursion pas terminale, c'est mal, puis j'ai attrapé une tendinite en devant taper ta combinaison de touches à 223 bpm.

Re: Bouh !

Posté par **omc** le 12/03/18 à 19:33. Évalué à 1 (+0/-0).

C'est une taptempoïte du métacarpe supérieur

Re: Bouh !

Posté par **Guillawme** (**page perso**) le 12/03/18 à 22:25. Évalué à 0 (+1/-1). Dernière modification le 12/03/18 à 22:30.

> *la récursion pas terminale, c'est mal*

Quelle solution proposes-tu pour l'utiliser avec Emacs Lisp qui n'optimise pas les appels récursifs terminaux ? :-)

Par curiosité, après avoir lu ton message j'ai essayé d'écrire une fonction factorielle récursive, et je ne peux même pas évaluer `(factorielle 2)` sans me prendre une erreur `Lisp nesting exceeds ‘max-lisp-eval-depth’`. Si j'augmente la valeur de cette limite, je bute sur `Variable binding depth exceeds max-specpdl-size` (et si j'augmente cette dernière, je reçois à nouveau la première erreur). J'avoue ne pas bien connaître Emacs Lisp (je m'en sers seulement pour configurer Emacs, ce qui consiste principalement à changer la valeur de variables… par réellement programmer), mais l'interpréteur semble déconseiller activement les solutions récursives.

Je me suis peut-être planté dans la définition, mais je ne crois pas car c'est la fonction récursive la plus facile à écrire :

```
(defun factorielle
    (n)
  "Factorielle récursive."
  (* n (factorielle (- n 1))))
```

Re: Bouh !

Posté par **Guillawme** (**page perso**) le 13/03/18 à 00:07. Évalué à 1 (+1/-0).

Ben je m'étais bel et bien planté… :P
La définition correcte c'est ça :

```
(defun factorielle (n)
  "Factorielle récursive."
  (if (= n 0)
      1
    (* n (factorielle (- n 1)))))
```

Avec la condition d'arrêt, ça fonctionne beaucoup mieux. :D Cela dit, on ne peut pas évaluer plus grand que `(factorielle 63)`.

Seconde erreur de ma part : ce n'est pas un appel récursif **terminal**… Je ne sais pas encore comment écrire ça.

Re: Bouh !

Posté par **kantien** le 13/03/18 à 10:37. Évalué à 6 (+4/-0). Dernière modification le 13/03/18 à 10:38.

> *Seconde erreur de ma part : ce n'est pas un appel récursif **terminal**… Je ne sais pas encore comment écrire ça.*

Avec un accumulateur ;-)
Je ne suis pas un expert en Elisp, mais ça doit ressembler à cela :

```
(defun fact-tailrec (n acc)
  "Factorielle récursive terminale"
  (if (= n 0)
     acc
     (fact-tailrec ((- n 1) (* n acc)))))

(defun factorielle (n)
   fact-tailrec (n 1))
```

--
Sapere aude ! Aie le courage de te servir de ton propre entendement. Voilà la devise des Lumières.

Erreur

Posté par **Wawet76** (**page perso**) le 12/03/18 à 14:16. Évalué à 2 (+0/-0).

Symbol's function definition is void: first

Ne répond pas à l'objectif

Posté par **jben** le 12/03/18 à 20:27. Évalué à 8 (+6/-0).

1. Il n'y a pas d'horizon de temps pris en compte. Seulement les `N` dernières frappes doivent être considérées.
2. Tu calcule entre chaque frappe une fréquence et tu retourne la moyenne des fréquences. Le comportement original est de calculer la fréquence comme l'inverse de la moyenne des périodes, et non comme la moyenne des inverses des périodes, ce n'est pas la même chose.

C'est bien caractéristique des adorateurs d'emacs ça, proposer un truc qui fait quelque chose, mais pas ce à quoi on s'attend raisonnablement.

Re: Ne répond pas à l'objectif

Posté par **Sacha Trémoureux** (**page perso**) le 12/03/18 à 21:44. Évalué à 4 (+3/-0).

> *C'est bien caractéristique des adorateurs d'emacs ça, proposer un truc qui fait quelque chose, mais pas ce à quoi on s'attend raisonnablement.*

C'est petit ça !

version 2 pré-béta

Posté par **omc** le 12/03/18 à 22:27. Évalué à 2 (+1/-0).

Compte tenu des commentaires ci-dessus, voici une version un peu moins approximative du bousin

```
(setq t0 nil)
(setq array (list))

(defun sum(list)
  (if (null list)
      0
    (+
     (nth 0 list)
     (sum (rest list))
     )
    )
  )

(defun mean(list)
  (/ (sum list) (length list))
  )


(defun tap ()
  "Record a tap"
  (interactive)
  (setq t1 (float-time()))
  (if t0 (push (- t1 t0) array))
  (if (> (length array) 9)
      (progn
    (message "BPM %S" (* 60 (/ 1. (mean array))))
    (setq array (reverse (cdr (reverse array))))
    ))
  (setq t0 t1))

(global-set-key "\C-x\C-s" 'tap)
```

Re: version 2 pré-béta

Posté par **Guillawme** (**page perso**) le 12/03/18 à 22:48. Évalué à 0 (+0/-0). Dernière modification le 12/03/18 à 22:51.

Ce serait quand même mieux d'attacher la fonction tap à une autre combinaison de touches que celle qui sert à enregistrer un fichier. Le préfixe `C-c` est là pour ça : il est réservé pour l'utilisateur (en supposant un Emacs de base ; dans la vraie vie, beaucoup de packages utilisent ce préfixe aussi…).

Ah, et il y a toujours un symbole non défini : `sum: Symbol’s function definition is void: rest`.

Re: version 2 pré-béta

Posté par **Frédéric Heulin** le 13/03/18 à 11:11. Évalué à 3 (+2/-0).

écrire une moyenne en emacs lisp : (source rosettacode.org)

```
(defun mean (lst)
    (/ (float (apply '+ lst)) (length lst)))

;; sample use
(mean '(1 2 3 4))
```

version 3 !

Posté par **omc** le 13/03/18 à 12:08. Évalué à 3 (+2/-0).

En tenant compte des différentes remarques, voici une nouvelle version

```
(setq t-start nil)
(setq t-end nil)
(setq t-list (list))
(setq t-data-num 9)

(defun tap ()
  "Record a tap"
  (interactive)
  (setq t-end (float-time()))
  (if t-start (push (- t-end t-start) t-list))
  (if (> (length t-list)  t-data-num)
      (progn
    (message "BPM %S" (* 60 (/ 1. (/ (float (apply '+ t-list)) (length t-list)))))
    (setq t-list (reverse (cdr (reverse t-list))))
    )
    (message "Result available in %S tap"(- t-data-num (length t-list)))
    )
  (setq t-start t-end)
  )

(global-set-key [C-f2] 'tap)
```

Re: version 3 !

Posté par **dark_star** le 13/03/18 à 12:22. Évalué à 4 (+2/-0).

on dira ce qu'on voudra, mais c'est le source le plus court du taptempo.
et java le plus long (lol)

Re: version 3 !

Posté par **mzf** (**page perso**) le 13/03/18 à 13:23. Évalué à 4 (+3/-0).

> *c'est le source le plus court du taptempo.*

Vu qu'il n'y a pas de paramètres d'entrées ni de régionalisation, je crois que le code golf en JavaScript est plus court :
https://linuxfr.org/users/wawet76/journaux/portage-de-taptempo-en-javascript#comment-1730795