

Boucles

Bruno Bouzy
1er septembre 2018

Ce document rassemble des exercices de C sur les instructions répétitives ou « boucles »: `for`, `do while` et `while`.

Exercice 1

Ecrire un programme `continue.c` qui affiche le menu suivant:

```
0: quitter
autre: continuer
```

On suppose que l'utilisateur tape un premier caractère différent de 'entrée', appelé caractère déterminant, puis 'entrée'. Si le caractère déterminant est 0, le programme se termine. Sinon le programme continue et le menu est affiché à nouveau. On écrira trois fois ce programme: en utilisant le `while`, en utilisant le `do while`, et en utilisant le `for`.

Quels sont les avantages et inconvénients de ces trois instructions ?

Exercice 1b

On souhaite rendre le programme `continue.c` plus robuste. On va donc écrire un programme `continue2.c` améliorant le programme `continue.c`. Une itération du programme `continue2.c` correspond à l'affichage du menu, à la lecture de un ou plusieurs caractères au clavier, le dernier caractère tapé étant toujours 'entrée'.

Successivement, on supposera que l'utilisateur peut taper :

- immédiatement 'entrée' auquel cas le programme continue (pas de caractère déterminant).
- des espaces avant de taper le caractère déterminant.
- des caractères quelconques entre le caractère déterminant et 'entrée'.

On utilisera des `do while`.

Exercice 2

Ecrire un programme `hellorepet.c` qui demande à l'utilisateur un nombre positif et affiche autant de fois `hello` sur une ligne de l'écran, en décalant le `hello` d'un espace vers la droite à chaque ligne:

```
entier positif ? 4
hello
 hello
  hello
   hello
```

On écrira trois fois ce programme: en utilisant le `while`, en utilisant le `do while`, et en utilisant le `for`.

Quels sont les avantages et inconvénients de ces trois instructions ?

Exercice 3

Ecrire un programme `notede0a20.c` demandant à l'utilisateur de taper une note entre 0 et 20 tant

que celui-ci ne tape pas un nombre entre 0 et 20, et le félicitant lorsqu'il a réussi. On écrira trois fois ce programme: en utilisant le `while`, en utilisant le `do while`, et en utilisant le `for`.

Exercice 4

Ecrire un programme `multiple3.c` demandant à l'utilisateur un nombre minimum, un nombre maximum et affichant les multiples de 3 compris entre ces deux nombres.

Exercice 5

Ecrire un programme `suite.c` demandant à l'utilisateur un nombre initial positif u_0 et affichant la suite telle que:

$$\begin{array}{ll} u_{n+1} = 3u_n + 1 & \text{si } u_n \text{ est impair,} \\ u_{n+1} = u_n / 2 & \text{si } u_n \text{ est pair.} \end{array}$$

Exercice 6

1) Ecrire un programme `alea.c` affichant x nombres aléatoires compris entre 0 et 20. On utilisera la fonction `srand` pour initialiser la suite aléatoire avec une graine donnée par l'utilisateur. On utilisera `rand` pour tirer effectivement un nombre aléatoire. On fera un `#include <stdlib.h>` afin d'utiliser les fonctions `srand` et `rand`.

2) Etendre programme `alea.c` pour qu'il affiche la moyenne et la variance des nombres aléatoires.

Exercice 7a

Ecrire un programme `secret.c` demandant à l'utilisateur un nombre maximal t , tirant un nombre secret au hasard compris entre 0 et t , et demandant à l'utilisateur de deviner ce nombre. Si l'hypothèse de l'utilisateur est la bonne, le programme félicite l'utilisateur et se termine. Si l'hypothèse de l'utilisateur est trop grande, le programme affiche `trop grand`, si elle est trop petite, le programme affiche `trop petit`. Tant que l'hypothèse est mauvaise, le programme demande une nouvelle hypothèse.

Améliorer le programme pour permettre à l'utilisateur de rejouer s'il le souhaite.

Exercice 7b

Ecrire un programme `secretinverse.c` demandant à l'utilisateur un nombre maximal t , tapé au clavier, et un nombre secret au hasard, compris entre 0 et t , restant secret dans la tête de l'utilisateur (ne pas le taper au clavier, le programme ne le connaît pas a priori). Le programme doit deviner le nombre secret de l'utilisateur par dichotomie. Répétitivement, le programme affiche une hypothèse. Si l'hypothèse du programme est la bonne, l'utilisateur tape `=` et se termine. Si l'hypothèse du programme est trop grande, l'utilisateur tape `>`, si elle est trop petite, l'utilisateur tape `<`. Tant que l'hypothèse est mauvaise, le programme propose une nouvelle hypothèse.

Améliorer le programme pour permettre à l'utilisateur de recommencer.