Boucles

Thomas Vincent (*thomasv0@cti.ecp.fr*)

6 novembre 2000

1 Boucles

1.1 Une boucle à quoi ca sert ?

Une boucle permet de répéter une instruction (ou une liste d'instructions) plusieurs fois.

Il y a principalement deux types de boucles
- Les boucles pour répéter une instruction un certain nombre de fois, il s'agit de la boucle **Pour**
- Les boucles pour répéter une instruction jusqu'à une condition d'arret, il s'agit des boucles **Tant que**

Le passage dans une boucle est appelé **itération**

1.2 Boucle pour

1.2.1 Definition

Les boucles **pour** permettent de répéter une instruction un nombre donné de fois.

Elle se caractérisent par le fait que l'on connait à l'avance le nombre d'itérations que l'on va devoir effectuer.

1.2.2 Propriete

A chaque instant, on connait le nombre d'itérations déja effectuées.

On connait aussi le nombre d'itérations restantes.

1.2.3 Syntaxe

Pour (variable) **allant de** (debut) **à** (fin) **faire**
instruction
Fin Pour

Une boucle commence par un **pour** et s'achève par un **fin pour**. La variable est appelée **variable de boucle** et il faut definir son minimum et son maximum

Exemple :
Si on suppose qu'une variable i entier a été declarée
Pour i **allant de** 0 **à** 10 **faire**
afficher(i)
Fin Pour

Cette boucle affichera successivement les nombres 0,1,...,10, on effectue donc 11 itérations

1.2.4 En C et C++

Pour i **allant de** debut **à** fin **faire**
instruction
Fin Pour
est traduit en C par

for ($i = debut$; $i < (fin + 1)$; $i++$) instruction ;

Les boucles **for** en C permettent bien d'autres choses, mais pour le moment nous nous limiterons à cette syntaxe.

1.3 Boucle tant que

1.3.1 definition

Les boucles tant que permettent d'effectuer des itérations tant qu'une certaine condition est verifiée.

On ne connait pas le nombre d'itérations à effectuer, mais à chaque itération, on vérifie si la condition est vraie ou fausse. dés que cette condition est fausse, on sort de la boucle.

1.3.2 Propriété

On sait qu'à la sortie de la boucle, la condition de boucle est fausse.

Attention : il faut s'assurer que les itérations permettent de modifier la valeur de la condition de boucle, si ce n'est pas le cas, la boucle ne s'arrète jamais.

Un exemple : si on effectue une boucle tant que dont le condition de poursuite est (a=0), si la variable a n'a pas de chance d'etre modifiée dans la boucle, il ne sera pas possible de sortir de cette boucle.

1.3.3 Syntaxe

Tant que condition **faire**
instruction
Fin tant que

Exemple :
n $\leftarrow$ 0
Tant que (n mod 21 $\neq$ 0) **faire**
n $\leftarrow$ n+15
Fin Tant Que

Cet algorithme va s'arreter des que n est un multiple de 21. A la sortie de l'algorithme, on possède donc la prorpiété suivante : n multiple de 21. Or comme n est toujours multiple de15, n est donc le premier nombre multiple de 15 qui est multiple de 21.

L'utilisation de la boucle **Tant que** est justifiée dans ce cas car on ne sait pas à l'avance, le nombre d'itérations effectuées.

1.3.4 En C et C++

Tant que (cond) **faire**
instruction
Fin tant que
est traduit en C par

while (cond) instruction ;

1.4 Pour et Tant que

1.4.1 Différence

Voici un petit exemple pour mettre en évidence la difference entre boucle pour et boucle tant que.

Supposons que vous vous trouviez en cours de sport et que le prof de sport, assis dans sa chaise longue, vous demande d'effectuer des tours de stade.
Il peut :

- Soit vous demander d'effectuer un certain nombre de tours de stade. L'algorithme de votre comportement sera donc basé sur une boucle **Pour**. De plus, à chaque instant, vous aurez conscience du (trop grand) nombre de tours qu'il vous reste à faire.
- Soit vous demander de courir jusqu'à que vous ne puissiez plus que ramper. La sortie du stade depend d'une condition : que vous soyez fatigué ou non. L'algorithme de votre comportement sera donc basé sur une boucle **Tant que** (de la forme, tant que (je ne suis pas fatigué) faire (je cours)). Vous ne saurez qu'une seule chose : lorsque vous arreterez de courir (si vous ne trichez pas), vous serez fatigué.

1.4.2 Quelle boucle choisir ?

Le choix de la boucle à utiliser dans un probléme se fait de la manière suivante.

Si on connait le nombre d'itération à effectuer dans la boucle, on utilisera une boucle **pour** .

Si la poursuite dans la boucle est dépendante d'une condition, on utilisera de préference une boucle **Tant que**