17. Langage : La boucle conditionnelle while( )

Dans certaines situations, il peut être utile d'effectuer une opération **tant qu'une certaine condition est vraie**. Par exemple lors de la réception de caractères sur le port série (tu me vois venir ?... non, sans blague ?!) ou d'un appui sur un bouton poussoir. La solution passe par ce que l'on pourrait appeler une boucle conditionnelle qui dit au microprocesseur :

- **tant que** (=while en anglais) telle condition est vraie, alors faire ceci
- sinon faire cela.

Cette boucle conditionnelle :

- est intéressante car elle permet , à la différence d'un boucle **for,** de répéter une tâche un nombre fois non défini à l'avance, répétition qui dépend d'une condition,
- est **à utiliser avec parcimonie, car elle peut bloquer un programme** (si la condition est toujours vraie) mais cet effet est parfois intéressant pour une exécution pas à pas ou un arrêt à un endroit précis du code.

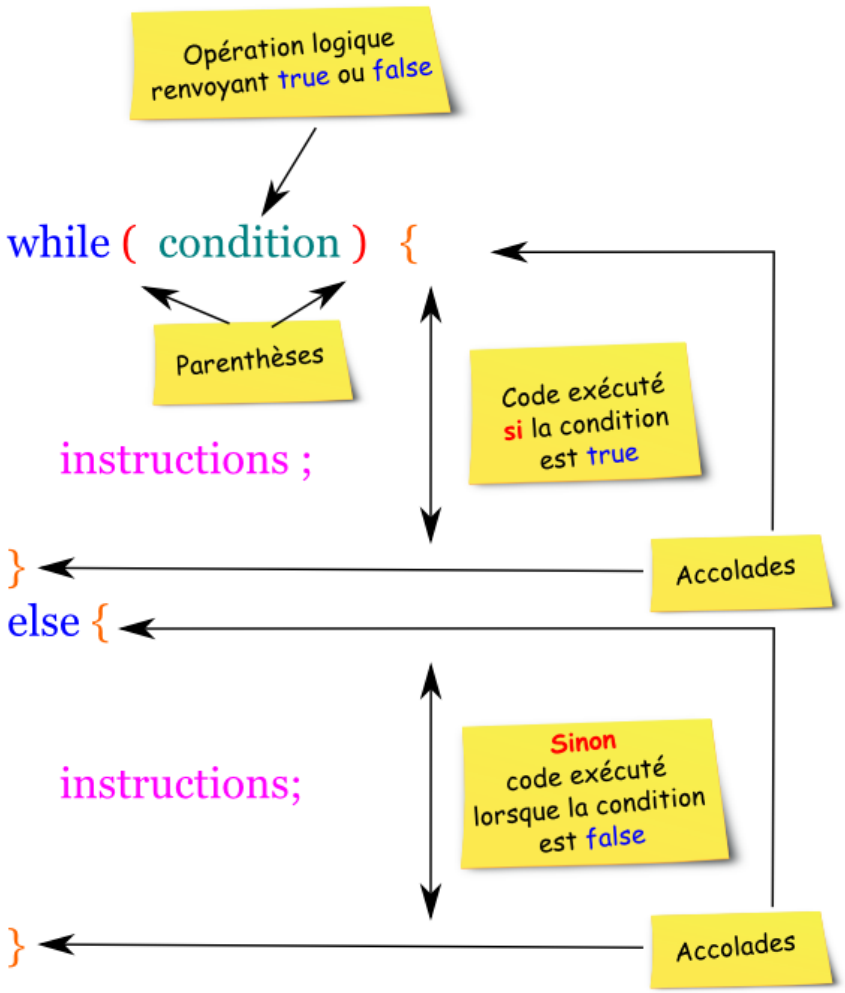
L'instruction utilisée pour une boucle conditionnelle « **tant que** » est l'instruction while ... else... (tant que... sinon...) que l'on écrit de la façon suivante (presque comme une condition if...else...):

- on commence par le mot clé while suivi de **la condition entre ( )**
- suivi des **{ } qui contiennent les instructions** à exécuter si la condition est vraie, chaque instruction devant être suivie d'un **;**
- en option, on peut compléter du mot clé else suivi des **{ }** qui contiennent les instructions à exécuter si la condition est fausse.

La condition devra être une opération logique :

- qui renverra une valeur de type boolean soit **true** (vrai) soit **false** (faux)
- on utilisera pour cela les **opérateurs logiques de comparaison**
- pour info, 0 est considéré comme false et toute valeur différente de 0 est considérée comme true (ainsi, if(1) est toujours vrai !)

Truc : while(1); permet de réaliser un point d'arrêt à n'importe quel endroit d'un programme (la condition est toujours vraie). Parfois utile !



18. Rappel : les opérateurs logiques et leur utilisation (utilisables avec if, while, else, etc..)

La condition de base utilisable est de tester si une variable vaut une certaine valeur :

- on écrira la condition sous la forme **variable==valeur** (**2 signes ==)**
- Par exemple on écrira if (variable==2) { instructions ;} ce qui veut dire « si la condition « variable vaut 2 » est vraie alors exécuter les instructions »

Ne pas confondre l'opération logique de test d'égalité == avec le signe = d'affectation :
c'est une erreur fréquente de débutant et même de programmeur expérimenté... !!

Les opérateurs logiques de comparaison

- x == y : VRAI si x est **égal à** y
- x **!=** y : VRAI si x est **différent de** y
- x < y : VRAI si x est **inférieur à** y
- x > y : VRAI si x est **supérieur à** y
- x <= y : VRAI si x est **inférieur ou égal à** y
- x >= y : VRAI si x est supérieur ou égal à y

Les opérateurs booléens (permettent d'enchaîner des conditions entre-elles)

- **&&** (ET logique) : VRAI seulement si les deux conditions sont VRAI

```
if (digitalRead(2) == HIGH  && digitalRead(3) == HIGH) { // lit l'état de 2 boutons poussoirs
  // ...
}
```

- || (OU logique) : VRAI si l'une des deux conditions est VRAI

```
if (x > 0 || y > 0) { // si x supérieur à 0 ou si y supérieur à 0
  // ...
}
```

- **!** (NON logique) : VRAI si l'opérande est FAUX – cas particulier : !x est VRAI chaque fois que x=0 (« tordu » mais pratique !)

```
if (!x) {
  // ...
}
```

19. Pour info : une variante : la boucle conditionnelle do... while

Juste pour votre information, car en pratique çà ne sert pas souvent, il existe une variante de la boucle while() : la boucle do.. while() (« faire... tant que.. »).

Description

La boucle **do / while** ("faire tant que" en anglais) fonctionne de la même façon que la boucle **while**, à la différence près que la condition est testée à la fin de la boucle, et par conséquent **la boucle do sera toujours exécutée au moins une fois**.

C'est subtil, mais çà peut parfois servir...

Syntaxe

```
do // faire...
{
    // bloc d'instruction

} while (condition); // tant que la condition est vraie
```

Exemple

```
do // faire...
{
  delay(50);          // attendre la stabilisation du capteur
  x = readSensors();  // lit la valeur de la tension du capteur

} while (x < 100); // ...tant que x est inférieur à 100
```