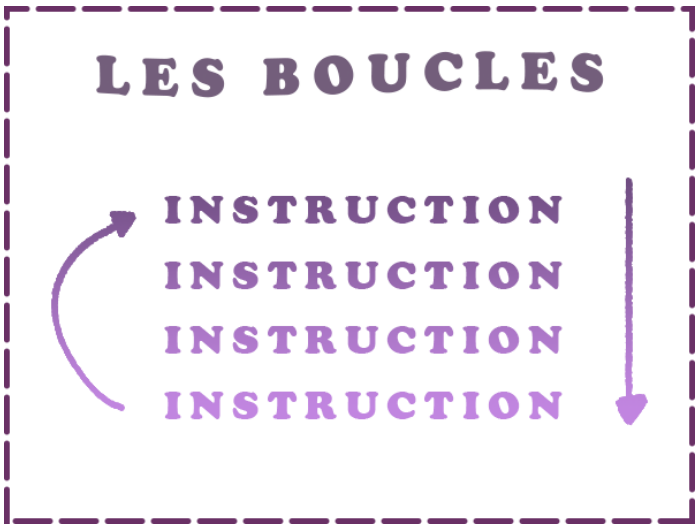
📝 Définition

Une boucle est une technique qui permet de répéter les mêmes instructions plusieurs fois. Il y a bien sûr plusieurs façons de réaliser des boucles :

- `while` ;
- `do... while` ;
- `for`.

Le principe est toujours le même :



Dans une boucle, l'ordinateur lit les instructions de haut en bas, puis, lorsqu'il arrive à la fin de la boucle, il repart à la première instruction ; il continue à faire cela autant de fois que spécifié.

⚠ Si on ne l'arrête pas, la machine continuera à faire tourner la boucle **à l'infini** ! Il faut prendre garde à ne pas bloquer l'ordinateur dans une boucle sans fin. Pour ce faire, on utilise les **conditions**. En gros, on dit à l'ordinateur : *reste dans cette boucle tant que* `X` *condition est vraie.*

Nous disions plus haut qu'il existe différentes façons de faire des boucles : sans plus tarder, voilà la première !

H2: Créer une boucle `while`

🖌 Syntaxe

On construit une boucle `while` de la façon suivante :

```
while (/* Condition */)
{
   // Instructions à répéter
}
```

📝 *While* signifique "tant que". On dit à l'ordinateur : "Tant que la condition est vraie, répète les instructions entre les accolades."

H5: Exemple 1 : tester la boucle `while` avec des instructions simples.

Exemple 1

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])

{
    int saisie = 0;

    while (saisie != 15) // tant que le nombre saisi est différent de 15...
    {
        printf("Saisissez le nombre 15\n");
        scanf("%d", &saisie);
    }

    printf("Merci !"); // une fois que le nombre saisi est 15, sortir de la boucle et afficher "Merci !"
}
```

Et le résultat de la boucle :

```
PS C:\Users\coral\Desktop\School\Exercices> .\while

Saisissez le nombre 15
14
Saisissez le nombre 15
85
Saisissez le nombre 15
46
Saisissez le nombre 15
15

Merci !
```

On voit donc, en faisant exprès de ne pas écouter les consignes, que la machine répète les instructions :

```c
    {
                printf("Saisissez le nombre 15\n");
        scanf("%d", &saisie);
    }
```

H5: Exemple 2 : faire en sorte qu'une boucle se répète un nombre donné de fois.

Imaginons qu'on ai besoin d'afficher une texte cinq fois ; on pour pour cela créer une variable `compteur` qu'on va incrémenter jusqu'à atteindre 5.

Voici le code qu'on rédige :

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])

{
    int compteur = 0; // on déclare la variable compteur et sa valeur initale

    while (compteur < 5) // tant que >|compteur est inférieur à 5...
    {
        printf("Boats ! \n"); // ... on affiche Boats !
        compteur++;           // et là on incrémente la valeur de >|compteur de 1 à chaque itération de la boucle.
    }

    return 0;
}
```

Et le résultat :

```
Boats !
Boats !
Boats !
Boats !
Boats !
```

📝 Explications

1. Notre variable `compteur` vaut `0` au début du programme.
2. En commençant la boucle, `compteur` vaut toujours `0`, mais à chaque itération, sa valeur augemente de 1 grâce à `compteur++` ;
3. Quand on arrive à la fin de la boucle, la machine vérifie si la condition `compteur < 5` est toujours vraie. Si oui, elle recommence la boucle et le fera jusqu'à ce que `compteur` vale `> 5`. La condition `compteur < 5` sera alors fausse et la boucle s'interrompra.

On peut afficher la valeur de compteur dans la boucle pour suivre son avancée :

```c
{
    int compteur = 0; // on déclare la variable compteur et sa valeur initale

    while (compteur < 5) // tant que >|compteur est inférieur à 5...
    {
        printf("Boats ! \n"); // ... on affiche Boats !
        compteur++;           // et là on incrémente la valeur de >|compteur de 1 à chaque itération de la boucle.
        printf("Compteur vaut %d\n", compteur);
    }

}
```

Ce qui donne :

```
Boats !
Compteur vaut 1
Boats !
Compteur vaut 2
Boats !
Compteur vaut 3
Boats !
Compteur vaut 4
Boats !
Compteur vaut 5
```

On voit bien que la boucle s'interrompt quand `compteur` atteint `5`.

⚠ **Attention aux boucles infinies ! **

Si vous lancer par erreur une boucle infinie, comme par exemple ceci :

```c
while (1)
{
    printf("Boucle infinie\n");
}
```

Le programme tournera littéralement à l'infini. Il faudra l'arrêter manuellement. On peut faire cela tout simplement en fermant la console, ou bien sous MacOS ou Linux, utiliser `ctrl + C`

H2: Créer une boucle `do... while`

On utilise moins souvent ce type de boucle. Elle diffère de `while` par le positionnement de la condition : au lieu d'être au début de la boucle, elle est à la fin.

Qu'est ce que ça change ?

Une boucle `while` peut très bien ne jamais s'excécuter si la condition s'avère fausse au départ.

Prenons l'exemple suivant :

```c
int compteur = 50;

do
{
    printf("Bienvenue sur OpenClassrooms !\n");
    compteur++;
} while (compteur < 10);
```

Ici, la condition d'exécution de la boucle (`compteur < 10`) est fausse d'entrée, puisque `compteur = 50`. Avec une condition `while`, on ne serait jamais rentré dans la boucle ; là, avec `do... while`, les instructions seront exécutées une fois.

Cette condition s'avère utilise quand on doit s'assurer que le programme excécute au moins une fois les instructions.

⚠ On met un `;` à la fin de `do... while`.

H2: Les boucles `for`

⚠ Les boucles `for` sont interdites à 42.

La boucle `for`, à la manière de `switch` pour les `if/else`, permets de rédiger les boucles de manière plus condensées.

Voilà une boucle `while` et une boucle `for` qui exécutent les mêmes instructions :

While :

```
int compteur = 0;

while (compteur < 10)
{
    printf("Bienvenue sur OpenClassrooms !\n");
    compteur++;
}
```

For

```
int compteur;

for (compteur = 0 ; compteur < 10 ; compteur++)
{
    printf("Bienvenue sur OpenClassrooms !\n");
}
```

📝 **Explications**

Les informations qui étaient à l'origine dans la boucle ont migré entre parenthèses après la condition. C'est là l'intêret de `for` ; elle permet de condenser entre parenthèses les instructions de la boucle.

⚠ On note que les instructions sont séparées par un `;`

En gros, les instructions se présentent ainsi :

```
for (compteur = 0 ; compteur < 10 ; compteur++)
for (initialisation de la variable ; déclaration de la variable ; incrémentation
    (ou toute autre instruction qui se répète en fin de boucle))
```

H2: En bref :

- Les **boucles** sont des **structures** qui permettent de répéter une série d'instructions tant qu'une condition n'est pas remplie.
- Il existe différents types de boucles, notamment `while`, `do... while` et `for` ; il faut juger au cas par cas de la pertinence de chacune.
- **A 42, on n'utilise pas** `for`. Selon OpenClassroom, c'est pourtant la condition qu'on utilise le plus.-