Chapitre 3 : Les Boucles

1. **La boucle Tant que**

Syntaxe :

```
Tant que condition faire
        instruction
Fin Tant que
```

En C et PHP :

```
while (condition)
{
        instruction ;
}
```

La boucle tant que exécute une ou plusieurs instructions autant de fois qu'une condition reste vérifiée. SI au départ la condition est non vérifiée, le bloc d'instructions est simplement sauté.
Dans le corps de la boucle, il faut prévoir une instruction qui fait évoluer la condition, sinon on aura une boucle infinie.

Le nombre d'itérations peut ne pas être connu à l'avance.

Représentation en organigramme :
voir le tableau

Exemple : afficher les nombres pairs compris entre 2 et 20.

```
Algo : nombres_pairs
Déclaration
        nb : entier
Début
        nb <- 2
        Tant que nb <= 20 faire
                afficher ("Nombre pair : ", nb)
                nb <- nb + 2
        Fin tant que
Fin nombres_pairs
```

Le schéma général de la boucle est le suivant :

```
Initialisation
Condition   <------|
Instruction         |
Evolution _______|
```

Traduction en C :

```
#include <stdio.h>
        int main ()
        {
                int nb ;
```

```
        nb = 2 ;
        while (nb <= 20)
        {
                printf("Nombre pair : %d", nb);
                nb = nb +2;
        }
        return 0;
}
```

2. **La Boucle Faire - Tant que**

Syntaxe :

```
Faire
        instruction
Tant que condition
```

En C et PHP :

```
do {
        instruction ;
}while (condition) ;
```

Cette boucle est un schéma inverse de la première. Le test de la condition est postérieur à l'exécution de la condition. Nous devons exécuter au moins une fois l'instruction avant de tester la condition. Comme la première boucle, le faire-tantque exécute le bloc d'instructions un nombre de fois qui peut être inconnu. Il faut toujours une instruction d'evolution dans le corps de la boucle.

Représentation en organigramme :

voir le tableau

Exemple : afficher les nombres pairs compris entre 2 et 20.

```
Algo : Nombres_pairs
Déclaration
        nb : entier
Début
        nb <- 2
        Faire
                Afficher ("Nombre pair :", nb )
                nb <- nb +2
        Tant que nb <= 20
Fin Nombres_pairs
```

Traduction en C :

```
#include <stdio.h>
        int main ()
        {
                int nb ;
                nb = 2;
                do{
                        printf("Nombre pair :%d", nb);
```

```
                    nb = nb +2;
            }while (nb <= 20);
            return 0;
    }
```

Le schéma général de la boucle est le suivant :

```
Initialisation
Evolution   <------|
Instruction        |
Condition _____|
```

3. La boucle Pour

Syntaxe :

```
Pour indice allant de VD à VF pas de PAS faire
        instruction
Fin Pour
```

VD : valeur de début
VF : valeur de fin
PAS : valeur d'incrémentation ou de décrémentation

EN C et PHP :

```
for (initialisation ; condition ; évolution )
{
        instruction ;
}
```

La boucle Pour est une boucle déterministe dont le nombre d'itérations est connu à l'avance : il est calculé avec les données suivantes : VD, VF et le PAS.

Représentation en organigramme :
voir le tableau

Exemple : Afficher les nombres pairs entre 2 et 20.

```
Algo : nombres_pairs
Déclaration
        nb : entier
Début
        Pour nb allant de 2 à 20 pas de 2 faire
                afficher("Nombre pair : ", nb)
        Fin pour
Fin nombre_pairs
```

Traduction en C :

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

```
int main ()
{
        int nb ;
        for (nb = 2; nb <=20 ; nb = nb +2)
        {
                printf("Nombre pair : %d", nb);
        }
        return 0;
}
```

Le schéma général de la boucle est le suivant :

```
Initialisation
Condition   <------|
Evolution          |
Instruction _____|
```