Chapitre 3 : Les boucles

1. La boucle TANT QUE

Cette boucle permet de répéter un ensemble d'instructions tant qu'une condition reste vraie. Elle teste la condition avant chaque itération.

```
Tant que condition faire
    Instruction
Fin tant que
```

Traduction en C++ :

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

2. La boucle FAIRE...TANT QUE

Cette boucle exécute d'abord le bloc d'instructions, puis vérifie la condition à la fin. Elle est donc toujours exécutée au moins une fois.

```
Faire
    Instruction
Tant que condition
```

Traduction en C++ :

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

3. La boucle POUR

Cette boucle est utilisée lorsque le nombre d'itérations est connu à l'avance. Elle permet d'initialiser, de tester et de faire évoluer une variable de contrôle.

```
Pour indice allant de VD à VF faire
    Instruction
Fin pour
```

Traduction en C++ :

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

Exercice 1 : Liste des diviseurs d'un entier

Écrire un algorithme, un programme en C, et un développement PHP permettant de saisir un entier et d'afficher la liste de ses diviseurs.

Algorithme :

```
Algo : diviseurs
Déclaration : nb, div : entier
Début
  Afficher("Donner un entier :")
  Saisir(nb)
  Pour div allant de 1 à nb faire
    Si nb modulo div = 0 Alors
      Afficher("Diviseur : ", div)
    Fin si
```

```
  Fin pour
Fin diviseurs
```

Traduction en C++ :

```
#include <stdio.h>
int main() {
  int nb, div;
  printf("Donnez un entier : ");
  scanf("%d", &nb;);
  for (div = 1; div <= nb; div++) {
    if (nb % div == 0) {
      printf("\nDiviseur : %d", div);
    }
  }
  return 0;
}
```

Exercice 2 : Nombres parfaits entre deux bornes

Écrire un algorithme, un programme C et un développement PHP permettant de déterminer tous les nombres parfaits compris entre deux bornes entières saisies par l'utilisateur. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs (sauf lui-même). Exemple : 28 est parfait car 1 + 2 + 4 + 7 + 14 = 28.