Programmation chapitre 3 : les itérations

1) définition

➤ Une **itération** est une **structure de contrôle** qui permet de **répéter plusieurs fois un bloc d'instructions**.

On utilise une itération lorsqu'on veut exécuter plusieurs fois une partie d'un algorithme **sans la réécrire**.

2) Types d'itérations

Type d'itération	Quand l'utiliser ?	Syntaxe générale

A bornes fixes	Le nombre de répétitions est **connu à l'avance.**	for (i=v1 ; i<=v2 ; i=i+1)
		bloc_instructions
		
		Plus généralement :
		
		for (*initialisation* ; *condition* ; *évolution*)
		bloc_instructions

Conditionnelles	Le nombre de répétitions **n'est pas connu mais seulement la condition d'arrêt.**	while (*condition*) bloc_instructions

a) Itérations à bornes fixes

➤ Principe

On connaît **exactement** combien de fois les instructions doivent être exécutées.

Boucle For = forme la + courante quand on connaît le nombre d'itérations.

Nb : les boucles peuvent être imbriquées (les unes dans les autres)

➤ Syntaxe

```
for (initialisation ; condition ; évolution)
    bloc_instructions
```

équivalente à :

```
initialisation ;
while (condition) {
    bloc_instructions
    évolution ;
}
```

3) ITÉRATIONS CONDITIONNELLES

➤ Principe

Lorsqu'on **ne connaît pas à l'avance le nombre de répétitions**, on utilise une **condition** pour décider de continuer ou non la boucle.

Deux formes possibles :

- WHILE : on teste la condition **avant** de répéter le traitement.
- DO...WHILE : on teste la condition **après** exécution (donc au moins une itération).

➤ Syntaxe

```
while (condition)
    bloc_instructions
```

```
do
    bloc_instructions
while (condition) ;
```

➤ Fonctionnement

- On teste d'abord la condition.
- Si la condition est fausse dès le départ, on sort directement de la boucle.
- Il faut toujours inclure dans le traitement une instruction qui fait évoluer la condition, pour éviter les boucles infinies.
- Si elle est vraie, on exécute le bloc, puis on reteste.

- On commence par exécuter les instructions contenues dans l'itération, puis teste la condition.
- Il faut toujours inclure dans le traitement une instruction qui fait évoluer la condition, pour éviter les boucles infinies.
- La condition décrit dans quels cas exécuter à nouveau le bloc d'instructions.

4) Opérations Raccourcies

Ces écritures condensent les opérations arithmétiques sur une même variable :

instruction	raccourci
i = i+1 ;	i++ ;
i = i-1 ;	i-- ;
i = i+j ;	i+=j ;
i = i-j ;	i-=j ;
i = i*j ;	i*=j ;
i = i/j ;	i/=j ;
i = i%j ;	i%=j ;

A retenir

- Itération = répétition contrôlée d'un bloc d'instructions.
- 2 types de boucles : à bornes fixes et conditionnelles.
- WHILE teste la condition **avant** d'exécuter.
- DO...WHILE exécute au - 1 fois avant de tester la condition.
- FOR : pratique pour les compteurs / parcours à pas régulier.
- Les **raccourcis (i++, +=, *=, ...)** rendent le code plus clair et concis.
- Bien vérifier la condition d'arrêt des boucles infinies.