Initiation à l'algorithmique

Structures répétitives « Les Boucles »

Mohamed MESSABIHI

mohamed.messabihi@gmail.com

Université de Tlemcen
Département d'informatique
1ère année MI

https://sites.google.com/site/informatiquemessabihi/

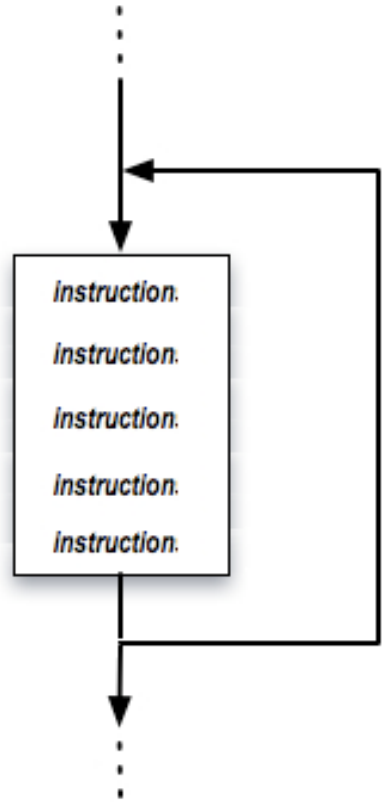
12 novembre 2014

Qu'est qu'une boucle ?

Une boucle est une structure de contrôle qui permet de répéter les mêmes instructions plusieurs fois.

On distingue types de boucles courantes en C :

1. **while**
2. **do... while**
3. **for**



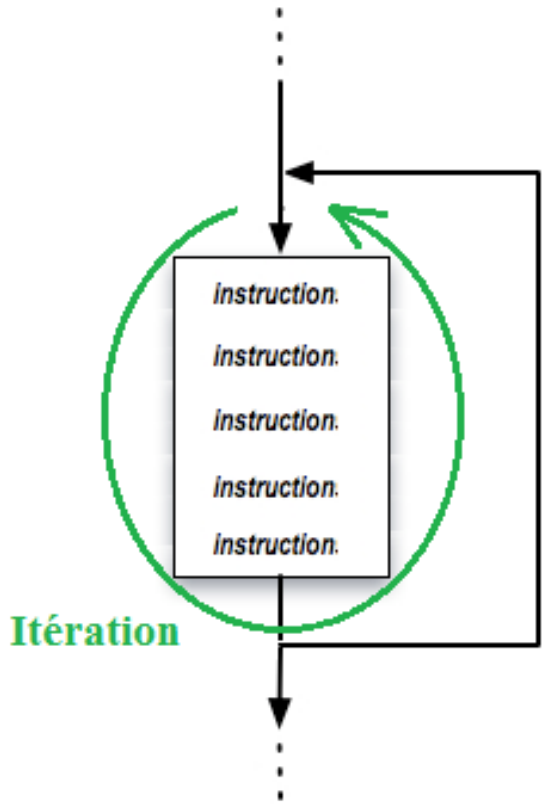
Le passage dans une boucle est appelé **itération**

Qu'est qu'une boucle ?

Une boucle est une structure de contrôle qui permet de répéter les mêmes instructions plusieurs fois.

On distingue types de boucles courantes en C :

1. **while**
2. **do... while**
3. **for**

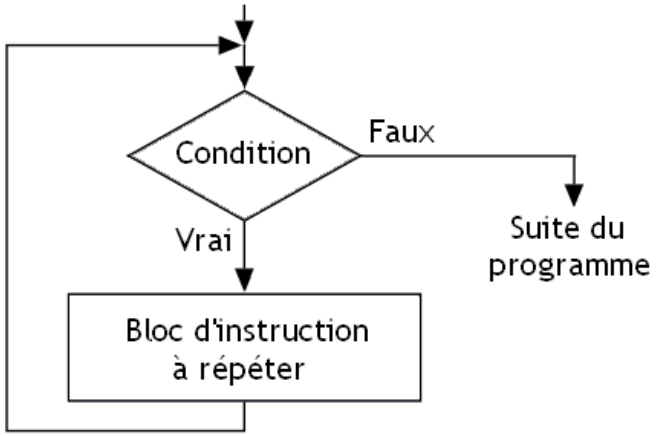


Le passage dans une boucle est appelé **itération**

La boucle « While »

Syntaxe :

```
while ( Condition )
{
    // Bloc d'instructions
}
```

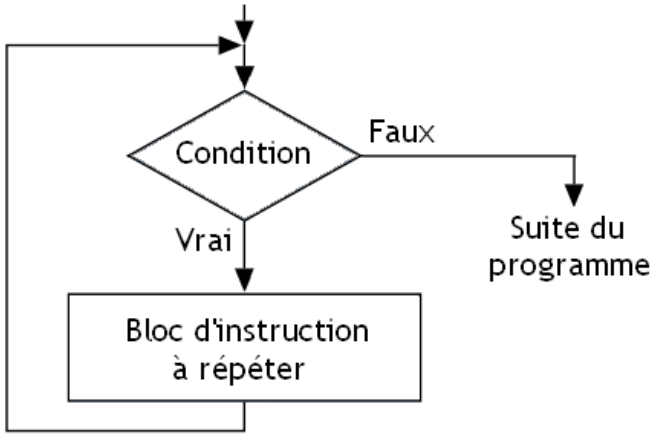


- la condition (dite condition de contrôle de la boucle) est évaluée avant chaque itération
- si la condition est vraie, on exécute le bloc d'instructions (corps de la boucle), puis, on retourne tester la condition. Si elle est encore vraie, on répète l'exécution,...
- si la condition est fausse, on sort de la boucle et on exécute l'instruction qui se trouve juste après l'accolade fermante « } ».

La boucle « While »

Syntaxe :

```
while ( Condition )
{
    // Bloc d'instructions
}
```

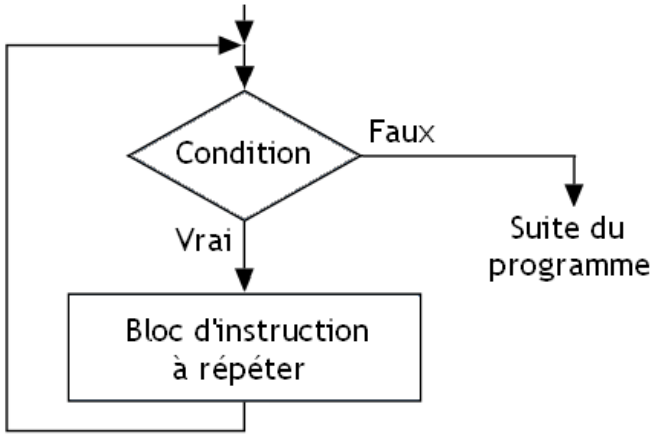


- la condition (dite condition de contrôle de la boucle) est évaluée avant chaque itération
- si la condition est vraie, on exécute le bloc d'instructions (corps de la boucle), puis, on retourne tester la condition. Si elle est encore vraie, on répète l'exécution,...
- si la condition est fausse, on sort de la boucle et on exécute l'instruction qui se trouve juste après l'accolade fermante « } ».

La boucle « While »

Syntaxe :

```
while ( Condition )
{
    // Bloc d'instructions
}
```



- la condition (dite condition de contrôle de la boucle) est évaluée avant chaque itération
- si la condition est vraie, on exécute le bloc d'instructions (corps de la boucle), puis, on retourne tester la condition. Si elle est encore vraie, on répète l'exécution,...
- si la condition est fausse, on sort de la boucle et on exécute l'instruction qui se trouve juste après l'accolade fermante « } ».

Exemple de boucle « While »

On souhaite écrire un programme qui permet de contrôler la saisie d'un entier positif.

Exemple :

```
int entierPositif = 0;

while (entierPositif <=0)
{
    printf("Tapez un entier positif ! ");
    scanf("%d", &entierPositif);
}
```

Répéter un certain nombre de fois

- On va pour cela créer une variable compteur qui vaudra 0 au début du programme
- Et que l'on va incrémenter à chaque itération.

Exemple :

```
int compteur = 0;

while (compteur < 10)
{
    printf("La variable compteur vaut %d \n", compteur);
    compteur++; // equivalent a compteur = compteur+1;
}
```

Une **incrémentation** consiste à ajouter 1 à la variable en faisant **var++ ;**.

Attention aux boucles infinies

- Le nombre d'itérations dans une boucle **while** n'est pas connu à l'avance. Il dépend de l'évaluation de la condition.
- Lorsque vous créez une boucle, assurez-vous toujours qu'elle peut s'arrêter à un moment ! Si la condition est toujours vraie, votre programme ne s'arrêtera jamais !

Exemple :

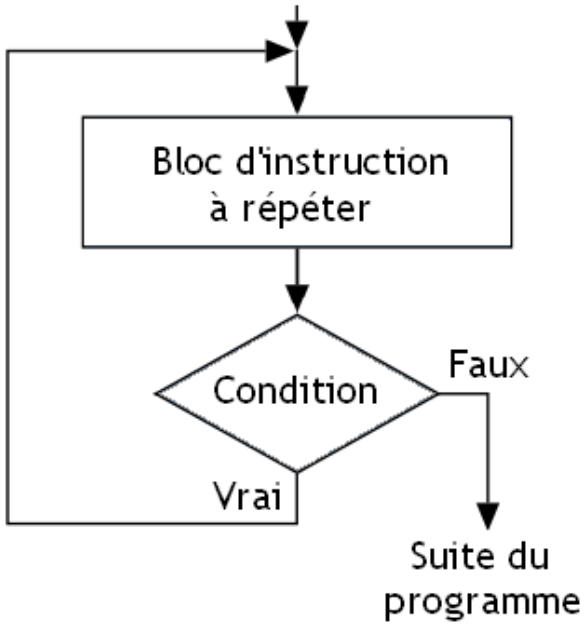
```
int compteur = 0;

while (compteur >= 0)
{
    printf("La variable compteur vaut %d \n", compteur);
    compteur++; // equivalent a compteur = compteur+1;
}
```

- Une des instructions du corps de la boucle doit absolument changer la valeur de condition de vrai à faux (après un certain nombre d'itérations), sinon le programme tourne indéfiniment

La boucle « do...while »

Syntaxe :

```
do
{
    // Bloc d'instructions
} while ( Condition );
```



- La boucle **do...while** est très similaire à while
- La seule chose qui change par rapport à while, c'est la position de la condition. Au lieu d'être au début de la boucle, la condition est à la fin.
- Cette boucle s'exécutera donc toujours au moins une fois

Exemple de boucle « do...while »

Exemple :

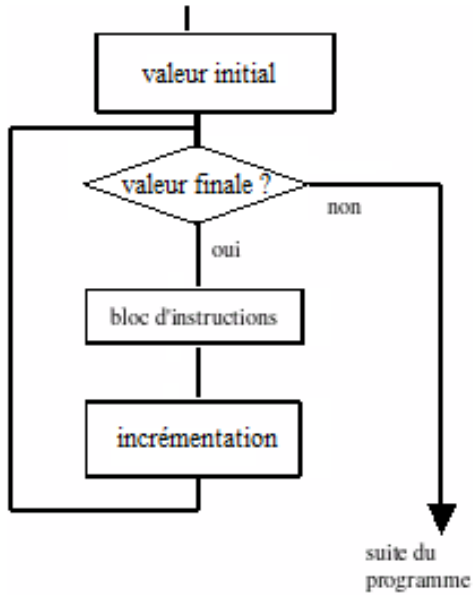
```
int compteur = 0;
int somme = 0;
do
{
    somme =+ compteur // equivalent a somme = somme+compteur
    compteur++;
}while (compteur <= 10);
```

Dans la boucle **« do...while »**, n'oubliez pas de mettre un point-virgule à la fin.

La boucle « For »

Syntaxe :

```
for (initialisation ; condition ; pas )
{
    // Bloc d'instructions
}
```



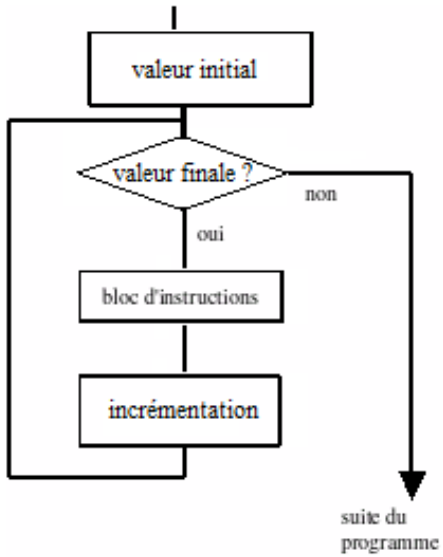
Il y a trois instructions condensées, chacune séparée par un point-virgule.

- La première est l'**initialisation** : cette première instruction est utilisée pour préparer notre variable compteur.
- La seconde est la **condition** : comme pour la boucle while, c'est la condition qui dit si la boucle doit être répétée ou non. Tant que la condition est vraie, la boucle **for** continue.
- Enfin, il y a l'**incrémentation** : cette dernière instruction est exécutée à la fin de chaque tour de boucle pour mettre à jour la variable compteur. par exemple).

Principe de la boucle « For »

Syntaxe :

```
int cpt;
for (cpt=initial; cpt<=finale; cpt=cpt+pas)
{
    printf("cpt vaut %d !\n", cpt);
}
```



1. La valeur **initiale** est affectée à la variable **cpt** ;
2. On compare la valeur du **cpt** et la valeur **finale** :
3. Si la **condition** est **fausse**, on sort de la boucle et on continue avec l'instruction qui suit l'accolade fermante.
4. Si la **condition** est **vrai** alors :
   - 4.1 les instructions de la boucle seront exécutées
   - 4.2 Ensuite, la valeur de **cpt** est **incrémentée** de la valeur du **pas** (sinon 1 par défaut).
   - 4.3 On recommence l'étape 2 : La comparaison entre cpt et finale est de nouveau effectuée, et ainsi de suite...

Exemple de la boucle « for »

Exemple :

```
int compteur;

for (compteur = 0 ; compteur < 10 ; compteur++)
{
    printf("La variable compteur vaut %d !\n", compteur);
}
```

La plus part du temps on fera une **incrémentation**, mais on peut aussi faire une **décrémentation (variable–)** ou encore n'importe quelle autre opération (**variable += 2**; pour avancer de 2 en 2 par exemple).

Attention !

Il est fortement déconseillé de modifier la valeur du compteur (et/ou la valeur de finale) à l'intérieur de la boucle. En effet, une telle action :

- perturbe le nombre d'itérations prévu par la boucle Pour
- présente le risque d'aboutir à une boucle infinie

Exemples :

```c
int compteur;

for (compteur = 0 ; compteur < 10 ; compteur++)
{
    printf("La variable compteur vaut %d !\n", compteur);
    compteur = compteur+2;
}

for (compteur = 9 ; compteur > 0 ; compteur--)
{
    printf("La variable compteur vaut %d !\n", compteur);
    compteur = compteur+2;
}
```

La boucle « For »Vs. La boucle « While »

La boucle Pour est un cas particulier de la boucle While (cas où le nombre d'itérations est connu et fixé) . Tout ce qu'on peut écrire avec For peut être remplacé par une boucle While (la réciproque n'est pas forcément vraie).

Exemple :

```
int compteur;
for (compteur = 0 ; compteur < 10 ; compteur++)
{
    printf("La variable compteur vaut %d !\n", compteur);
}
// est equivalent a :
int compteur = 0;
while (compteur < 10)
{
    printf("La variable compteur vaut %d !\n", compteur);
    compteur++;
}
```

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Les boucles imbriqués

Le bloc d'instructions d'une boucle peut contenir lui même une autre boucle. C'est ce qu'on appelle des boucles imbriquées

Exemple :

```
int i;
int j=1;

for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d et j=%d!\n", i, j);
    j++;
   }
}
```

Historique d'exécution

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Quelle boucle puisse-je utiliser pour mon programme ?

Utilisez la structure qui reflète le mieux l'idée du programme que vous voulez réaliser :

- Si le bloc d'instructions ne doit pas être exécuté si la condition est fausse, alors utilisez while ou for.
- Si le bloc d'instructions doit être exécuté au moins une fois, alors utilisez do - while.
- Si le nombre d'exécutions du bloc d'instructions est connu à l'avance alors utilisez for.
- Si le bloc d'instructions doit être exécuté aussi longtemps qu'une condition extérieure est vraie alors utilisez while.

Le choix entre for et while n'est souvent qu'une question de préférence ou d'habitudes.

Quelle boucle puisse-je utiliser pour mon programme ?

Utilisez la structure qui reflète le mieux l'idée du programme que vous voulez réaliser :

- Si le bloc d'instructions ne doit pas être exécuté si la condition est fausse, alors utilisez while ou for.
- Si le bloc d'instructions doit être exécuté au moins une fois, alors utilisez do - while.
- Si le nombre d'exécutions du bloc d'instructions est connu à l'avance alors utilisez for.
- Si le bloc d'instructions doit être exécuté aussi longtemps qu'une condition extérieure est vraie alors utilisez while.

Le choix entre for et while n'est souvent qu'une question de préférence ou d'habitudes.

Quelle boucle puisse-je utiliser pour mon programme ?

Utilisez la structure qui reflète le mieux l'idée du programme que vous voulez réaliser :

- Si le bloc d'instructions ne doit pas être exécuté si la condition est fausse, alors utilisez while ou for.
- Si le bloc d'instructions doit être exécuté au moins une fois, alors utilisez do - while.
- Si le nombre d'exécutions du bloc d'instructions est connu à l'avance alors utilisez for.
- Si le bloc d'instructions doit être exécuté aussi longtemps qu'une condition extérieure est vraie alors utilisez while.

Le choix entre for et while n'est souvent qu'une question de préférence ou d'habitudes.

Quelle boucle puisse-je utiliser pour mon programme ?

Utilisez la structure qui reflète le mieux l'idée du programme que vous voulez réaliser :

- Si le bloc d'instructions ne doit pas être exécuté si la condition est fausse, alors utilisez while ou for.
- Si le bloc d'instructions doit être exécuté au moins une fois, alors utilisez do - while.
- Si le nombre d'exécutions du bloc d'instructions est connu à l'avance alors utilisez for.
- Si le bloc d'instructions doit être exécuté aussi longtemps qu'une condition extérieure est vraie alors utilisez while.

Le choix entre for et while n'est souvent qu'une question de préférence ou d'habitudes.

Quelle boucle puisse-je utiliser pour mon programme ?

Utilisez la structure qui reflète le mieux l'idée du programme que vous voulez réaliser :

- Si le bloc d'instructions ne doit pas être exécuté si la condition est fausse, alors utilisez while ou for.
- Si le bloc d'instructions doit être exécuté au moins une fois, alors utilisez do - while.
- Si le nombre d'exécutions du bloc d'instructions est connu à l'avance alors utilisez for.
- Si le bloc d'instructions doit être exécuté aussi longtemps qu'une condition extérieure est vraie alors utilisez while.

Le choix entre for et while n'est souvent qu'une question de préférence ou d'habitudes.

Quelle boucle puisse-je utiliser pour mon programme ?

Utilisez la structure qui reflète le mieux l'idée du programme que vous voulez réaliser :

- Si le bloc d'instructions ne doit pas être exécuté si la condition est fausse, alors utilisez while ou for.
- Si le bloc d'instructions doit être exécuté au moins une fois, alors utilisez do - while.
- Si le nombre d'exécutions du bloc d'instructions est connu à l'avance alors utilisez for.
- Si le bloc d'instructions doit être exécuté aussi longtemps qu'une condition extérieure est vraie alors utilisez while.

Le choix entre for et while n'est souvent qu'une question de préférence ou d'habitudes.

Exemples pour conclure ...

Exemple avec while :

```
int N;          /* nombre de donnees */
int NOMB;       /* nombre courant    */
int I;          /* compteur */
long SOM;       /* la somme   des nombres entres */
double PROD; /* le produit des nombres entres */

printf("Nombre de donnees : ");
scanf("%d", &N);
SOM=0; PROD=1; I=1;
while(I<=N)
    {
     printf("%d. nombre : ", I);
     scanf("%d", &NOMB);
     SOM  += NOMB;
     PROD *= NOMB;
     I++;
    }

printf("La somme   des %d nombres est %ld \n", N, SOM);
printf("Le produit des %d nombres est %.0f\n", N, PROD);
```

Un exemple pour conclure ...

Exemple avec do...while :

```
int N;          /* nombre de donnees */
int NOMB;       /* nombre courant    */
int I;          /* compteur */
long SOM;       /* la somme   des nombres entres */
double PROD; /* le produit des nombres entres */

printf("Nombre de donnees : ");
scanf("%d", &N);

SOM=0;
PROD=1;
I=1;
do
   {
    printf("%d. nombre : ", I);
    scanf("%d", &NOMB);
    SOM  += NOMB;
    PROD *= NOMB;
    I++;
   } while(I<=N);
```

Un exemple pour conclure ...

Exemple avec for :

```
int N;          /* nombre de donnees */
int NOMB;       /* nombre courant    */
int I;          /* compteur */
long SOM;       /* la somme   des nombres entres */
double PROD;    /* le produit des nombres entres */


printf("Nombre de donnees : ");
scanf("%d", &N);


for (SOM=0, PROD=1, I=1 ; I<=N ; I++)
    {
     printf("%d. nombre : ", I);
     scanf("%d", &NOMB);
     SOM  += NOMB;
     PROD *= NOMB;
    }
```