

Itérations : approfondissements et exemples

Exemples d'autres formes de boucles `for`

```
for(int p(0); p < 10; p += 2) {
```

...
la variable `p` prendra les valeurs de 0, 2, 4, 6, 8 (`p += 2` est équivalent à `p = p + 2`)

```
for(int k(10); k > 0; --k) {
```

...
la variable `k` prendra les valeurs 10, 9, 8 ... jusqu'à 1

```
for(int i(0); i >= 0; ++i) {
```

...
la condition est toujours vraie (du moins dans le principe).

La boucle est répétée indéfiniment et la variable `i` prendra toutes les valeurs positives que le type `int` peut représenter

Boucles infinies

La boucle `for` peut ne pas s'arrêter, ce qui se produit quand la condition est toujours vraie. Plusieurs causes sont possibles:

1. On s'est trompé sur la condition:

Par exemple:

```
for(int i(0); i > -1; ++i) { // !!!
```

2. On s'est trompé sur l'incrémentation:

```
for(int i(0); i < 10; ++j) { // !!!
```

`j` est incrémenté au lieu de `i`, `i` garde donc toujours la valeur 0, et la boucle ne s'arrête pas.

Pas de point-virgule (;) à la fin de l'instruction `for`

Les instructions suivantes n'affichent qu'une seule fois la chaîne "bonjour":

```
for(int i(0); i < 10; ++i;  
    cout << "bonjour" << endl;
```

Le point-virgule seul est considéré comme une instruction (qui ne fait rien).

Le corps de la boucle est donc constitué de cette instruction qui ne fait rien:

```
for(int i(0); i < 10; ++i)  
;  
cout << "bonjour" << endl;
```

Attention aux accolades

```
for(int i(0); i < 5; ++i)
    cout << "i = " << i << endl;
    cout << "Bonjour" << endl;
```

affiche:

```
i = 0
i = 1
i = 2
i = 3
i = 4
Bonjour
```

Interprétation:

```
for(int i(0); i < 5; ++i)
    cout << "i = " << i << endl;
cout << "Bonjour" << endl;
```

Evitez de modifier une variable compteur à l'intérieur d'une boucle for

```
for(int i(0); i < 10; ++i) {
    ...
    if (...)
        --i; // !!!
}
```

1. Ça ne fera sans doute pas ce que vous voulez : n'oubliez pas que la boucle for, de son côté, incrémente la variable i.
2. Un relecteur risque de ne pas s'apercevoir que la variable est modifiée également à l'intérieur de la boucle, et de ne pas comprendre le fonctionnement.

Moyenne de 4 notes

Sans boucle for, en utilisant 5 variables:

```
double note1;
cout << "Entrez la note numero 1" << endl;
cin >> note1;

double note2;
cout << "Entrez la note numero 2" << endl;
cin >> note2;

double note3;
cout << "Entrez la note numero 3" << endl;
cin >> note3;

double note4;
cout << "Entrez la note numero 4" << endl;
cin >> note4;

double somme(note1 + note2 + note3 + note4);
cout << "Moyenne = " << somme / 4 << endl;
```

Sans boucle for, en n'utilisant que 2 variables:

```
double note, somme(0.0);

cout << "Entrez la note numero 1" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 2" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 3" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 4" << endl;
cin >> note;
somme = somme + note;

cout << "Moyenne = " << somme / 4 << endl;
```

Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double note, somme(0.0);

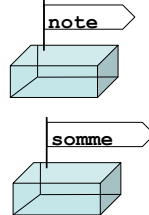
cout << "Entrez la note numero 1" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 2" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 3" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 4" << endl;
cin >> note;
somme = somme + note;

cout << "Moyenne = " << somme / 4 << endl;
```



Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double note, somme(0.0);

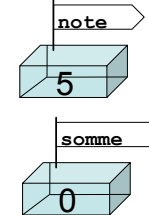
cout << "Entrez la note numero 1" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 2" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 3" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 4" << endl;
cin >> note;
somme = somme + note;

cout << "Moyenne = " << somme / 4 << endl;
```



Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double note, somme(0.0);

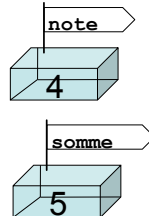
cout << "Entrez la note numero 1" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 2" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 3" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 4" << endl;
cin >> note;
somme = somme + note;

cout << "Moyenne = " << somme / 4 << endl;
```



Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double note, somme(0.0);

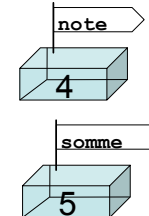
cout << "Entrez la note numero 1" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 2" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 3" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 4" << endl;
cin >> note;
somme = somme + note;

cout << "Moyenne = " << somme / 4 << endl;
```



Pour vérifier le programme précédent, supposons que l'utilisateur entre les notes 5, 4, 6 et 4:

```
double note, somme(0.0);

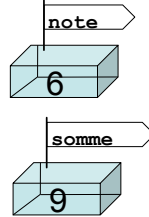
cout << "Entrez la note numero 1" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 2" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 3" << endl;
cin >> note;
somme = somme + note;

cout << "Entrez la note numero 4" << endl;
cin >> note;
somme = somme + note;

cout << "Moyenne = " << somme / 4 << endl;
```



Même programme en utilisant une boucle `for`.

```
double note, somme(0.0);

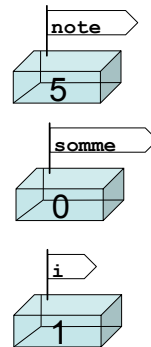
for(int i(1); i <= 4; ++i) {
    cout << "Entrez la note numero " << i << endl;
    cin >> note;
    somme = somme + note;
}

cout << "Moyenne = " << somme / 4 << endl;
```

```
double note, somme(0.0);

for(int i(1); i <= 4; ++i) {
    cout << "Entrez la note numero " << i << endl;
    cin >> note;
    somme = somme + note;
}

cout << "Moyenne = " << somme / 4 << endl;
```



Comment modifier le code pour laisser l'utilisateur choisir le nombre de notes ?

```
double note, somme(0.0);

for(int i(1); i <= 4; ++i) {
    cout << "Entrez la note numero " << i << endl;
    cin >> note;
    somme = somme + note;
}

cout << "Moyenne = " << somme / 4 << endl;
```

```
double note, somme(0.0);
int nombre_de_notes;

cout << "Entrez le nombre de notes" << endl;
cin >> nombre_de_notes;

for(int i(1); i <= nombre_de_notes; ++i) {
    cout << "Entrez la note numero " << i << endl;
    cin >> note;
    somme = somme + note;
}

cout << "Moyenne = " << somme / nombre_de_notes << endl;
```

Il y a un **bug!**

```
double note, somme(0.0);
int nombre_de_notes;

cout << "Entrez le nombre de notes" << endl;
cin >> nombre_de_notes;

for(int i(1); i <= nombre_de_notes; ++i) {
    cout << "Entrez la note numero " << i << endl;
    cin >> note;
    somme = somme + note;
}

cout << "Moyenne = " << somme / nombre_de_notes << endl;
```

Une **solution**:

```
double note, somme(0.0);
int nombre_de_notes;

cout << "Entrez le nombre de notes" << endl;
cin >> nombre_de_notes;

if (nombre_de_notes > 0) {
    for(int i(1); i <= nombre_de_notes; ++i) {
        cout << "Entrez la note numero " << i << endl;
        cin >> note;
        somme = somme + note;
    }

    cout << "Moyenne = " << somme / nombre_de_notes << endl;
}
```

Boucles imbriquées

Reprenons l'exemple précédent de la table de multiplication par 5:

```
for(int i(1); i <= 10; ++i) {
    cout << "5 multiplie par " << i << " vaut " << 5 * i << endl;
}
```

Supposons qu'on veuille maintenant afficher toutes les tables de multiplication, de 2 à 10.

Il suffit de mettre la boucle précédente dans une autre boucle, et de remplacer le 5 par...ce qu'il faut.

Boucles imbriquées

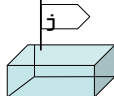
```
for(int j(2); j <= 10; ++j) {  
    for(int i(1); i <= 10; ++i) {  
        cout << "5 multiplie par " << i << " vaut " << 5 * i << endl;  
    }  
}
```

affiche 9 fois la table de multiplication par 5

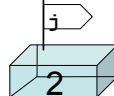
Boucles imbriquées

```
for(int j(2); j <= 10; ++j) {  
    for(int i(1); i <= 10; ++i) {  
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;  
    }  
}
```

affiche la table de multiplication par 2, puis par 3, jusque 10.



```
→ for(int j(2); j <= 10; ++j) {  
    cout << "Table de multiplication par " << j << ":" << endl;  
    for(int i(1); i <= 10; ++i) {  
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;  
    }  
}
```



```
for(int j(2); j <= 10; ++j) {  
→ cout << "Table de multiplication par " << j << ":" << endl;  
    for(int i(1); i <= 10; ++i) {  
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;  
    }  
}
```

Table de multiplication par 2:
|

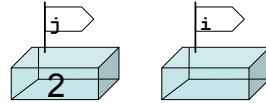
```

for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

```

Table de multiplication par 2:

1



```

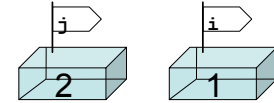
for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

```

Table de multiplication par 2:

2 multiplie par 1 vaut 2

1



```

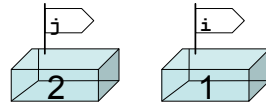
for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

```

Table de multiplication par 2:

2 multiplie par 1 vaut 2

1



```

for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

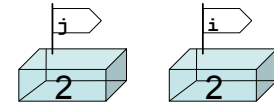
```

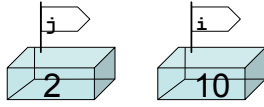
Table de multiplication par 2:

2 multiplie par 1 vaut 2

2 multiplie par 2 vaut 4

1





```

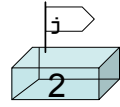
for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
|

```



```

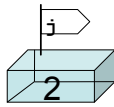
for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
|

```



```

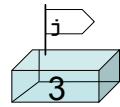
for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
|

```



```

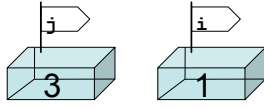
for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
Table de multiplication par 3:
|

```

```

for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

```

```

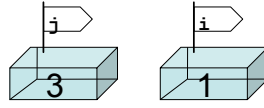
Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
Table de multiplication par 3:

```

```

|

```



```

for(int j(2); j <= 10; ++j) {
    cout << "Table de multiplication par " << j << ":" << endl;
    for(int i(1); i <= 10; ++i) {
        cout << j << " multiplie par " << i << " vaut " << j * i << endl;
    }
}

```

```

Table de multiplication par 2:
2 multiplie par 1 vaut 2
2 multiplie par 2 vaut 4
...
2 multiplie par 10 vaut 20
Table de multiplication par 3:
3 multiplie par 1 vaut 3

```

```

|

```