Itérations : introduction

Il y a 3 structures de contrôle:

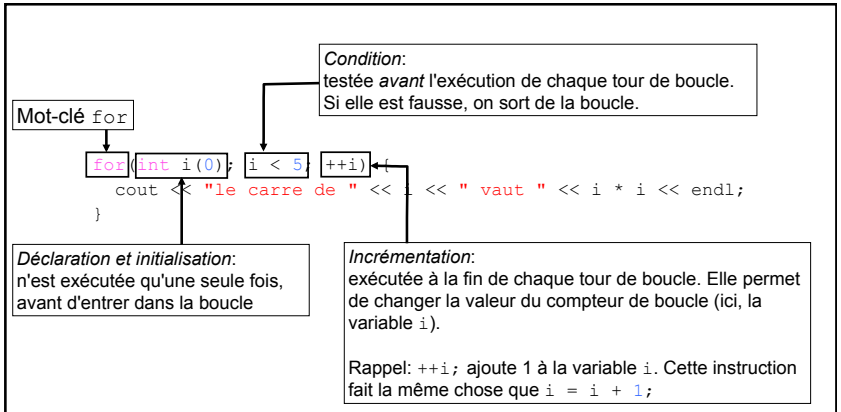
- les branchements conditionnels,
- les **itérations**, et
- les boucles conditionnelles.

La boucle `for`

Une boucle `for` permet de répéter un nombre donné de fois la même série d'instructions.
Supposons qu'on veuille afficher les carrés des 5 premiers entiers.
On peut utiliser le code suivant:

```
for(int i(0); i < 5; ++i) {
  cout << "le carre de " << i << " vaut " << i * i << endl;
}
```

```
le carre de 0 vaut 0
le carre de 1 vaut 1
le carre de 2 vaut 4
le carre de 3 vaut 9
le carre de 4 vaut 16
```



L'initialisation, la condition, et l'incrémentation sont séparées par des points-virgules

```
for(int i(0); i < 5; ++i) {
  cout << "le carre de " << i << " vaut " << i * i << endl;
}
```

Corps de la boucle:
Bloc d'instructions qui seront exécutées à chaque tour de boucle.

Comme pour le `if`, les accolades ne sont obligatoires que si plusieurs instructions doivent être répétées.

Si il n'y a qu'une seule instruction, on peut ne pas utiliser d'accolades:

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

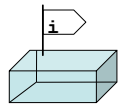
Mais, toujours comme pour le `if`, il est conseillé de garder les accolades:

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

Pas-à-pas

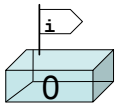
```
for(int i(0); i < 5; ++i) {
  cout << "le carre de " << i << " vaut " << i * i << endl;
}
```

Ce qui s'affiche dans la fenêtre Terminal:



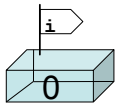

```
for(int i(0); i < 5; ++i) {
  cout << "le carre de " << i << " vaut " << i * i << endl;
}
```

Ce qui s'affiche dans la fenêtre Terminal:




```
for(int i(0); i < 5; ++i) {
    cout << "le carre de " << i << " vaut " << i * i << endl;
}
```

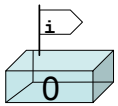
Ce qui s'affiche dans la fenêtre Terminal:




```
for(int i(0); i < 5; ++i) {
    cout << "le carre de " << i << " vaut " << i * i << endl;
}
```

Ce qui s'affiche dans la fenêtre Terminal:

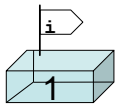
```
le carre de 0 vaut 0
```




```
for(int i(0); i < 5; ++i) {
    cout << "le carre de " << i << " vaut " << i * i << endl;
}
```

Ce qui s'affiche dans la fenêtre Terminal:

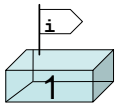
```
le carre de 0 vaut 0
```




```
for(int i(0); i < 5; ++i) {
    cout << "le carre de " << i << " vaut " << i * i << endl;
}
```

Ce qui s'affiche dans la fenêtre Terminal:

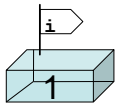
```
le carre de 0 vaut 0
```




```
    for(int i(0); i < 5; ++i) {
→       cout << "le carre de " << i << " vaut " << i * i << endl;
    }
```

Ce qui s'affiche dans la fenêtre Terminal:

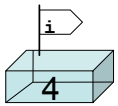
```
le carre de 0 vaut 0
le carre de 1 vaut 1
```




```
→   for(int i(0); i < 5; ++i) {
        cout << "le carre de " << i << " vaut " << i * i << endl;
    }
```

Ce qui s'affiche dans la fenêtre Terminal:

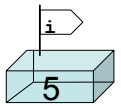
```
le carre de 0 vaut 0
le carre de 1 vaut 1
```




```
→   for(int i(0); i < 5; ++i) {
        cout << "le carre de " << i << " vaut " << i * i << endl;
    }
```

Ce qui s'affiche dans la fenêtre Terminal:

```
le carre de 0 vaut 0
le carre de 1 vaut 1
le carre de 2 vaut 4
le carre de 3 vaut 9
le carre de 4 vaut 16
```




```
→   for(int i(0); i < 5; ++i) {
        cout << "le carre de " << i << " vaut " << i * i << endl;
    }
```

Ce qui s'affiche dans la fenêtre Terminal:

```
le carre de 0 vaut 0
le carre de 1 vaut 1
le carre de 2 vaut 4
le carre de 3 vaut 9
le carre de 4 vaut 16
```

```
for(int i(0); i < 5; ++i) {
  cout << "le carre de " << i << " vaut " << i * i << endl;
}
```

Le programme continue en exécutant les instructions après la boucle.

La variable `i` n'est déclarée que pour l'intérieur de la boucle.

Elle ne peut pas être utilisée à l'extérieur de la boucle.

Ce qui s'affiche dan

```
le carre de 0 va
le carre de 1 va
le carre de 2 vaut 4
le carre de 3 vaut 9
le carre de 4 vaut 16
```

Syntaxe de l'instruction `for`

```
for(déclaration_et_initialisation; condition; incrémentation) {
  bloc
}
```

- Si la condition ne devient jamais fausse, les instructions dans la boucle sont répétées indéfiniment !

Affichage d'une table de multiplication

Dans le programme suivant, la même ligne ou presque est répétée 10 fois:
Une constante prend les valeurs de 1 à 10.

```
cout << "Table de multiplication par 5:" << endl;

cout << "5 multiplie par 1 vaut " << 5 * 1 << endl;
cout << "5 multiplie par 2 vaut " << 5 * 2 << endl;
cout << "5 multiplie par 3 vaut " << 5 * 3 << endl;
cout << "5 multiplie par 4 vaut " << 5 * 4 << endl;
cout << "5 multiplie par 5 vaut " << 5 * 5 << endl;
...
```

→ il faut utiliser une boucle `for` pour éviter cette répétition.

Affichage d'une table de multiplication

On peut remplacer:

```
cout << "5 multiplie par 1 vaut " << 5 * 1 << endl;
cout << "5 multiplie par 2 vaut " << 5 * 2 << endl;
cout << "5 multiplie par 3 vaut " << 5 * 3 << endl;
cout << "5 multiplie par 4 vaut " << 5 * 4 << endl;
cout << "5 multiplie par 5 vaut " << 5 * 5 << endl;
...
```

par

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

La variable `i` prend ici les valeurs de 1 à 10.

Que s'affiche-t-il quand on exécute le code :

```
for(int i(0); i < 5; ++i) {
  cout << i;
  if (i % 2 == 0) {
    cout << "p";
  }
  cout << " ";
}
cout << endl;
```

A: `0p 1 2p 3 4p`

B: `0p 1 2 3 4`

C: `0 1 2p 3 4`

D: `0p 1p 2p 3p 4p`

?