Les blocs

Les blocs

En C++, les instructions peuvent être regroupées en **blocs**.

Les blocs sont identifiés par des délimiteurs de début et de fin : **{** et **}**

Exemple:

```
{
  int i;
  double x;

  cout << "Valeurs pour i et x : " << endl;
  cin >> i >> x;
  cout << "Vous avez entré : i = " << i << i
       << ", x = " << x << endl;
}
```

Les blocs

Les blocs ont en C++ une grande autonomie.

Ils peuvent contenir leurs propres déclarations et initialisation de variables:

```
if (i != 0) {
  int j(0);

  ...
  j = 2 * i;
  ...
}
// A partir d'ici, on ne peut plus utiliser j
```

Notion de portée

- Les variables déclarées à l'intérieur d'un bloc sont appelées **variables locales** (au bloc). Elles ne sont accessibles qu'à l'intérieur du bloc.

```
if (i != 0) {
  int j(0);

  ...
  j = 2 * i;
  ...
}
// A partir d'ici, on ne peut plus utiliser j
```

Notion de portée

- Les variables déclarées à l'intérieur d'un bloc sont appelées **variables locales** (au bloc). Elles ne sont accessibles qu'à l'intérieur du bloc.

- Les variables déclarées en dehors de tout bloc (même du bloc `main(){}`) sont appelées **variables globales** (au programme).
  Elles sont accessibles dans l'ensemble du programme.

Bonnes pratiques:
1. Ne jamais utiliser de variables globales (sauf peut-être pour certaines constantes).
2. Déclarer les variables au plus près de leur utilisation.

Notion de portée

Déclarer les variables au plus près de leur utilisation

Par exemple, si la variable `j` n'est pas utilisée après la condition,

écrivez:

```
if (i != 0) {
  int j(0);

  ...
  j = 2 * i;
  ...
}
```

plutôt que:

```
int j(0);
if (i != 0) {

  ...
  j = 2 * i;
  ...
}
```

Notion de portée

La **portée** d'une variable, c'est l'ensemble des lignes de code où cette variable est accessible, autrement dit où elle est définie, existe, a un sens.

```
if (i != 0) {
  int j(0);

  ...
  j = 2 * i;
  ...
  if (j != 2) {
    int k(0);

    ...
    k = 3 * i;
    ...
  }
  ...
}
```

Règles de résolution de portée

```
if (i != 0) {
  int j(0);

  ...
  j = 2 * i;
  ...
  if (j != 2) {
    int j(0);

    ...
    j = 3 * i;
    ...
  }
  ...
}
```

En cas d'ambiguïté, c'est-à-dire quand plusieurs variables de portées différentes portent le même nom:

la variable « la plus proche » est choisie

Bonnes pratiques:
Évitez d'utiliser plusieurs fois le même nom de variables, sauf peut-être pour des variables locales aux boucles, comme `i`, `j`, ...

Portée : cas des itérations

La déclaration d'une variable à l'intérieur d'une itération est une déclaration **locale au bloc de la boucle**, et aux deux instructions de test et d'incrément:

```
for(int i(0); i < 5; ++i) {
   cout << i << endl;
}
// A partir d'ici, on ne peut plus utiliser ce i
```

Exemple à ne pas suivre !

```
#include <iostream>
using namespace std;

int main()
{
  int i(120);

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

  cout << i << endl;

  return 0;
}
```