Le C++ moderne par le developpement de jeux

-

La déclaration de variables

Nicolas Koenig

Il est possible de déclarer ses variables de différentes façons en C++. Nous avons déjà vu en session 4 trois syntaxes pour déclarer ses variables.
D'autres syntaxes existent et il peut être utile de les connaître si vous les rencontrez un jour dans un code source qui n'est pas le vôtre.

Ce document récapitule l'ensemble des syntaxes possibles pour la déclaration de variables, leurs cas d'utilisation, leurs avantages et leurs inconvénients. Si vous ne travaillez qu'en solo et sur du code propre à vous, les syntaxes à utiliser sont de toutes façons décrites dans les sessions vidéos.

NODATEK
formation & développement informatique

1. Les declarations vues en session 4

Dans l'ordre, nous avons vu pour le moment les 3 syntaxes suivantes :

1.1 Déclaration typée avec initialisation uniforme

La syntaxe avec les accolades se nomme "initialisation uniforme" :

```
type nomVariable{valeur};
```

Pour information, ne pas mettre de valeur revient à initialiser à zéro :

```
type nomVariable{};
```

Ici, nomVariable vaut 0.

Inconvénient : la valeur peut être implicitement convertie en le type donné. Autrement dit, bien qu'un avertissement est lancé, il n'y a pas d'erreur si le type ne peut pas contenir la valeur.

Cas d'utilisation : jamais (privilégier la déclaration auto)

1.2 Déclaration typée sans initialisation

```
type nomVariable;
```

L'ordinateur réserve un espace mémoire mais ne donne pas de valeur à la variable. Il est donc impossible de savoir la valeur que contient notre variable avec cette syntaxe.

Cas d'utilisation : jamais

1.3 Déclaration auto avec initialisation uniforme

```
auto nomVariable{valeur};
```

Cas d'utilisation : à privilégier
(sauf dans les trois cas de la partie 3 sur les déclarations utiles vues plus tard)

Avantages :
- Pas besoin de connaître le type de sa variable
- Impossible d'oublier d'initialiser sa variable
- Aucune conversion implicite possible (donc pas de perte d'information)

2. Les declarations possibles mais inutiles

2.1 Déclaration typée avec parenthèses

```
type nomVariable(valeur);
```

Cette déclaration est équivalente à la déclaration typée avec initialisation uniforme, à la différence que sans valeur, le code ne compile pas :

```
type nomVariable();
```

Ici, nomVariable ne vaut pas 0 (contrairement à l'initialisation uniforme) : le code ne fonctionne pas.

Cas d'utilisation : jamais (préférez utiliser les accolades)

2.2 Déclaration typée avec le signe =

```
type nomVariable = valeur;
```

Cette syntaxe est encore une fois équivalente, mais peut porter à confusion : ici le signe égal réalise une initialisation et non une affectation de nouvelle valeur.

Cas d'utilisation : jamais (privilégiez la déclaration uniforme)

2.3 Déclaration typée avec = {}

```
type nomVariable = {valeur};
```

Équivalente à la déclaration avec initialisation uniforme en moins concise.

Cas d'utilisation : jamais (privilégiez la déclaration uniforme)

2.4 Déclaration auto avec parenthèses

```
auto nomVariable(valeur);
```

Équivalente à la déclaration auto avec initialisation uniforme.

Cas d'utilisation : jamais (privilégiez la déclaration uniforme)

3. Les declarations utiles vues plus tard

3.1 Déclaration auto avec le signe =

```
auto nomVariable = valeur;
```

Équivalente à la déclaration auto avec initialisation uniforme.

Cas d'utilisation : lorsque la valeur est le résultat d'une fonction, car cette syntaxe est alors plus lisible :

```
auto nomVariable = calculerMoyenne(3,7);
```

est plus lisible que :

```
auto nomVariable {calculerMoyenne(3,7)};
```

(Nous reverrons cette syntaxe dans la session sur les fonctions)

3.2 Déclaration auto avec = type{}

```
auto nomVariable = type{valeur};
```

Équivalente à la déclaration typée avec initialisation uniforme.

Cas d'utilisation : Inutile pour les types simples, cette notation est à utiliser pour déclarer des objets de types plus complexes.
(Nous reverrons cette syntaxe dans la session sur la programmation orientée objet)

3.3 Déclaration auto avec = type()

```
auto nomVariable = type(valeur);
```

Équivalente à la déclaration typée, mais permet de construire un objet sans l'initialiser.

Cas d'utilisation : Inutile pour les types simples, cette notation est parfois utile pour créer un objet en appelant un constructeur particulier pour les types qui ont des constructeurs à liste d'initialisation.

Exemple : pour construire un vector de 5 éléments sans initialiser leurs valeurs :

```
auto monVector = std::vector(5);
```

(Nous reverrons cette syntaxe dans la session sur la programmation orientée objet)