mettre ce chapitre aussi tôt ?

Portées locales et globales des variables

Portée locale et durée de vie

Dans les chapitres précédents, les variables sont locales, elles sont détruites en sortant du bloc :

```
{
    int const i {};
    // utilisation de i
} // i est détruit ici
```

Avec une portée locale, plus facile de suivre et comprendre le rôle et l'évolution des valeurs. Le comportement est déterministe, on sait quand sont créées les variables et quand elles sont détruites.

Il est possible de créer des blocs, uniquement destinés à contrôler la durée de vie des variables :

```
int main() {
    int i { 123 };
    {
        int const j { i + 456 };
        i += j;
    } // j est détruit ici
    cout << i << endl;
}
```

- Portée : partie du code dans laquelle une variable est liée à un identifiant, un nom. C'est à dire, de sa déclaration à sa sortie de bloc.

- durée de vie : de la création de la variable à la destruction.

Actuellement, nous n'avons vu que des variables dont la durée de vie et la portée correspondent. Mais, vous verrez, par la suite, qu'il est possible de créer des objets qui ont une durée de vie différente de leur portée (au travers des objets alloués dynamiquement sur le Tas).

Il existe également des variables qui ont une portée globale (visible dans tout le code) ou une valeur statique (la même pour tout le code), mais leur utilisation doit se faire avec beaucoup de précautions. En effet, la lecture du code est rendue plus difficile, surtout sur de gros projets (on ne peut savoir quand une variable est modifiée) et il peut vite devenir complexe de modifier le traitement de ces variables.

L'utilisation de variables non locales doit toujours se faire avec des pincettes, en connaissant parfaitement tous les tenants et les aboutissants de cette décision.

Variables globales

portée dans toutes l'application, pas dans un bloc.

```
int i {};

void f() {
    cout << i << endl;
}

int main() {
    ++i;
    f();
    ++i;
    f();
}
```

Variables statiques

valeur est conservée

```
void f() {
   static int i {};
   ++i;
   cout << i << endl;
}

int main() {
    f();
    f();
    f();
}
```

affiche :

```
1
2
3
```

Chapitre précédent	Sommaire principal	Chapitre suivant

Cours, C++