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 local, plus facile de suivre et comprendre le rôle et l'évolution des valeurs. Le comportement est déterministe, on sait quand sont crées les variables et quand elles sont détruites.

Possible de créer des blocs, juste 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 pendant laquelle une variable est accessible. ie à partir du moment où elle est déclarée et détruite au moment où l'on sort du bloc.

- durée de vie : quand un objet est créé et quand il est détruit

Actuellement, on a vu que des variables dont la durée de vie et la portée sont identique. Vous verrez pas la suite qu'il est possible de créer des objets qui ont une durée de vie différentes de la portée (objets dynamique sur le Tas).

Possible d'avoir une portée globale ou une valeur statique, mais rend plus difficile la lecture du code, surtout sur de gros projets (on ne peut savoir quand une variable est modifiée)

L'utilisation de variables non locales doit toujours se faire après avoir vérifié que c'était la seule solution

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++