Ce chapitre est un peu spéciale. Le but est de vous présenter des techniques... qu'il ne faut pas utiliser ! Ou tout au moins, qu'il faut fortement limiter leur utilisation. La raison est que cela produit généralement une dette technique et diminue la qualité du code (moins maintenable, moins évolutif et plus difficilement testable).

Une dette technique est le surcoût de travail nécessaire lorsque vous faites des mauvais choix dans votre code. La conséquence, par exemple, est qu'au lieu d'implémenter une fonctionnalité en une semaine, vous l'implémenterez en deux semaines. Plus le temps passe et plus les problèmes augmenteront dans la dette technique, jusqu'à ce que vous corriger ces problèmes.

Donner des explications détaillées sur pourquoi les globaux et assimilés posent problème est assez difficile, puisque cela demande de l'expérience et du recul, de préférence sur des projets conséquents. Il va falloir dans un premier temps accepter cette règle : "interdit d'utiliser les globaux !"

Variables globales et statiques

Variables globales

Dans les chapitres precedants, vous avez vu uniquement les variables locales, qui etait definie a partir de leur declaraiton et n'etait plus valide a la fin du bloc.

```
int main() {
    int i { 123 };                    // déclaration de i
    {
        int j { 456 };                // déclaration de j
        std::cout << j << std::endl;  // j est valide
    }                                 // destruction de j
    std::cout << i << std::endl;      // i est valide
```

```
}                                       // destruction de i
```

Une variable *globale* est une variable dont la portée n'est pas limitée à un bloc, mais peut s'étendre a tout le programme.

Un premier exemple simple est la déclaration d'une variable en dehors de la fonction `main`.

```
int i {};

int main() {
    ++i;
    std::cout << i << std::endl;
    ++i;
    std::cout << i << std::endl;
}
```

affiche :

```
1
2
```

Variables statiques

Une variable *statique* est une variable dont la 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