# Éléments sur `const` et `constexpr`

Michel Simatic et Loïc Joly

Télécom SudParis

23 avril 2026

# `const`

- `const` permet de déclarer une variable qui ne peut plus être modifiée après son initialisation, la rendant *read-only*. Cela garantit qu'une variable ne changera pas durant le fonctionnement du programme. Cela améliore la qualité du code, sa lisibilité, sa maintenabilité.
- ⇒ Utilisez `const` dès que vous le pouvez !
- *East* `const` *versus West* `const` (choisissez votre religion)

```
int const i1 = 5; // East const
const int i2 = 6; // West const
```

- `const` s'applique à l'élément qui le précède… sauf s'il est au début (*West* `const`) auquel cas il s'applique à l'élément suivant.

# Exemples de `const`

```cpp
int const i1 = 5; // Entier constant initialisé à 5
const int i2 = 5; // Idem

int* p1; // Rien n'est constant (ni le pointeur, ni l'entier pointé)

const int* p2; // Pointeur non constant sur un entier constante
int const* p3; // Idem

int *const p4; // Pointeur constant sur un entier non constant

int const* const p5; // Pointeur constant sur un entier constant
const int* const p6; // Idem

const int reponse_a_la_vie_etc = knows_douglas_adams == true ? 42 : 0; // Initialise une variable constante avec un test
```

# `const` dans une classe

Soit la classe `Personne` :

```
class Personne {
public:
  Personne(std::string nom, int age);
  std::string getPersonne() const; // getPersonne() est une méthode qui ne modifie aucune variable d'instance
private
  const std::string nom;
  int age;
}
```

Le constructeur suivant ne compile pas. Pourquoi ?

```
Personne::Personne(const std::string nom_, const int age_) {
  nom = nom_;
  age = age_;
}
```

En revanche, celui-ci compile (et, cerise sur le gâteau, n'a pas besoin de différencier les noms) :

```
Personne::Personne(const std::string nom, const int age)
  : nom{nom} // Le premier `nom` est la variable d'instance ; le deuxième est le paramètre
  , age{age}
{}
```

# `const` *versus* `constexpr`

```
const     int i1 = 5; // Vraie variable dont la valeur ne changera pas durant l'exécution
constexpr int i2 = 6; // Variable calculée à la compilation
```

- Utiliser `constexpr` améliore l'exécution du code ⇒ Le préférer à `const`
- `constexpr` peut concerner une fonction.
- `constexpr` est obligatoirement en début de ligne.

# Questions ?

sachant qu'on a vu :

1. `const` (qui est à utiliser dès que possible !)
2. `const` dans une classe
3. `const` *versus* `constexpr` (qui est à privilégier)