Chapitre 2 : Syntaxe élémentaire en langage C++

Objectifs pédagogiques :

- Maîtriser la structure syntaxique de base du C++
- Comprendre et utiliser les différents types de données fondamentaux
- Savoir déclarer et manipuler des variables et constantes
- Maîtriser les opérateurs et leur priorité pour écrire des expressions complexes
- Acquérir les bonnes pratiques de documentation du code

1. Structure générale d'un programme C++

Avant d'entrer dans le détail de la syntaxe, rappelons la structure typique d'un programme C++ :

```
// 1. Directives de préprocesseur
#include <iostream>

Using namespace std ;

// 2. Fonction principale
int main() {

    // 3. Instructions
    cout << "Bonjour ! " << endl;

    // 4. Retour de fonction
    return 0;
}
```

2. Commentaires : Documenter son code

En automatique, la documentation est cruciale car vos programmes seront probablement maintenus ou modifiés par d'autres ingénieurs.

2.1. Commentaires sur une ligne

Exemple : `// Ceci est un commentaire sur une seule ligne`

2.2. Commentaires multilignes

```
/*
 * Ceci est un commentaire
 * sur plusieurs lignes
 * Très utile pour décrire un algorithme complexe
 */
```

3. Mots-clés et identificateurs

3.1. Mots-clés réservés

Le C++ possède environ 90 mots-clés qui ont une signification spéciale et ne peuvent pas être utilisés comme noms de variables. Exemple : `int, void, return, namespace, …`

3.2. Règles de nommage (Identificateurs)

- Doivent commencer par une lettre ou underscore `_`
- Peuvent contenir lettres, chiffres, underscore
- Sensibles à la casse : `Vitesse` ≠ `vitesse`
- Éviter de commencer par `_` (réservé pour les implémentations système)

4. Variables et constantes

4.1. Déclaration de variables

Syntaxe : type nom = valeur_initiale;

```
int compteur;                    // Déclaration seule (valeur indéterminée)
double position = 0.0;           // Déclaration avec initialisation
float vitesse = 2.5f;            // 'f' pour float
bool moteur_en_marche = false;   // Booléen
char code_erreur = 'A';          // Caractère (entre quotes simples)

// Déclaration multiple
int a = 5, b = 10, c = 15;
double x, y, z;                   // Déclarés mais pas initialisés
```

4.2. Constantes

```
// Constante traditionnelle (C-style)
const double GRAVITE = 9.81;

// Constante expression (C++11)
constexpr double PI = 3.141592653589793;
```

5. Types fondamentaux

5.1. Types numériques

Type	Taille typique	Plage	Utilisation
`bool`	1 octet	`true` ou `false`	États logiques
`char`	1 octet	-128 à 127 ou 0 à 255	Caractères, petits entiers

Type	Taille typique	Plage	Utilisation
`int`	4 octets	-2.147×10^{9} à 2.147×10^{9}	Compteurs, indices
`short`	2 octets	-32768 à 32767	Économie mémoire
`long`	4 ou 8 octets	dépend du système	Grands entiers
`float`	4 octets	$\pm3.4\times10^{-38}$ à $\pm3.4\times10^{38}$	Précision simple
`double`	8 octets	$\pm1.7\times10^{-308}$ à $\pm1.7\times10^{308}$	Précision double

6. Opérateurs

6.1. Opérateurs arithmétiques

Opérateur	Description	Exemple	Résultat
`+`	Addition	`5 + 3`	`8`
`-`	Soustraction	`5 - 3`	`2`
`*`	Multiplication	`5 * 3`	`15`
`/`	Division	`5 / 2`	`2` (entiers)
`/`	Division	`5.0 / 2`	`2.5` (flottants)
`%`	Modulo (reste)	`5 % 2`	`1`

6.2. Opérateurs de comparaison

Opérateur	Signification	Exemple	Résultat
`==`	Égal à	`5 == 3`	`false`
`!=`	Différent de	`5 != 3`	`true`
`<`	Inférieur à	`5 < 3`	`false`

Opérateur	Signification	Exemple	Résultat
`>`	Supérieur à	`5 > 3`	`true`
`<=`	Inférieur ou égal	`5 <= 5`	`true`
`>=`	Supérieur ou égal	`5 >= 3`	`true`

6.3. Opérateurs logiques

Opérateur	Signification	Exemple	Résultat
`&&`	ET logique	`(5>3) && (2<4)`	`true`
`||`	OU logique	`(5<3) || (2<4)`	`true`
`!`	NON logique	`!(5<3)`	`true`

6.4. Opérateurs d'incrémentation/décrémentation

```
int compteur = 5;

// Post-incrémentation (utilise puis incrémente)
int a = compteur++;  // a = 5, compteur = 6

// Pré-incrémentation (incrémente puis utilise)
int b = ++compteur;  // compteur = 7, b = 7

// Équivalent à :
compteur = compteur + 1;
```

6.5. Opérateurs d'affectation composée

```
x += 5.0;      // x = x + 5    -> 15.0
x -= 3.0;      // x = x - 3    -> 12.0
x *= 2.0;      // x = x * 2    -> 24.0
x /= 4.0;      // x = x / 4    -> 6.0

int y = 10;
y %= 3;        // y = y % 3    -> 1
```