

C01 Introduction

Programmation en langage C++

BTS CIEL 1^{ère} année

Lycée Louis Rascol, Albi



Release : v1.2 (2023-09-02) [je.serrand](#)

 [ciel-ir-rascol/cpp-cours](https://github.com/ciel-ir-rascol/cpp-cours)

Sommaire

1. Pourquoi apprendre le C++ ?
2. Historique
3. Processus de construction du programme
4. Structure d'un programme C++
5. Les commentaires
6. Les directives de préprocesseur
7. La fonction main()
8. L'espace de nom
9. Les flux d'entrée / sortie

Un langage populaire

- Beaucoup d'applications sont encore écrites en C++ de nos jours
- Arrive souvent dans les premiers langages sur les classements
- Une communauté très active sur GitHub, Stack Overflow ...

[Schedule a demo](#)

| Aug 2023 | Aug 2022 | Change | Programming Language | Ratings | Change |
|----------|----------|--------|--|---------|--------|
| 1 | 1 | |  Python | 13.33% | -2.30% |
| 2 | 2 | |  C | 11.41% | -3.35% |
| 3 | 4 | ▲ |  C++ | 10.63% | +0.49% |
| 4 | 3 | ▼ |  Java | 10.33% | -2.14% |
| 5 | 5 | |  C# | 7.04% | +1.64% |

Un langage pertinent

- **OS** : Windows, Linux, Mac OSX ...
- **Logiciels** : Adobe (Photoshop, Illustrator), Bases de données (MySQL, MongoDB)
- **Entreprises** : Amazon, Microsoft, Apple, Paypal, Google ...
- **Applications** : Réalité Virtuelle, IA, Telecom, Réseaux ...

Un langage puissant

- Rapide, Flexible, Adaptable, Portable
- Paradigmes : Procédural, Orienté Objet
- Beaucoup de langages de plus haut niveau utilisent C ou C++
- 😞 Langage complexe, courbe d'apprentissage abrupte

Des opportunités de carrière

- Compétence très recherchée
- Des salaires supérieurs
- Des aptitudes utilisables dans tous les autres langages de programmation
- Facilités à apprendre de nouveaux langages quand C++ est acquis

Sommaire

1. Pourquoi apprendre le C++ ?
2. **Historique**
3. Processus de construction du programme
4. Structure d'un programme C++
5. Les commentaires
6. Les directives de préprocesseur
7. La fonction main()
8. L'espace de nom
9. Les flux d'entrée / sortie

Le C++ classique

- **Début 1970** : Dennis Ritchie invente le C
- **1979** : Bjarne Stroustrup crée *C with Classes*
- **1983** : Changement du nom pour C++
- **1989** : Première sortie commerciale
- **1998** : C++98
- **2003** : C++03



Dennis Ritchie en 2011



Bjarne Stroustrup en 2010

Le C++ moderne

À partir de la version 11 : **C++ Moderne**, nouvelles fonctionnalités, simplifications. . .

- **2011** : C++11
- **2014** : C++14
- **2017** : C++17
- **2020** : C++20

 **Dans ce cours nous utiliserons les standards du C++ moderne**

Sommaire

1. Pourquoi apprendre le C++ ?
2. Historique
- 3. Processus de construction du programme**
4. Structure d'un programme C++
5. Les commentaires
6. Les directives de préprocesseur
7. La fonction main()
8. L'espace de nom
9. Les flux d'entrée / sortie

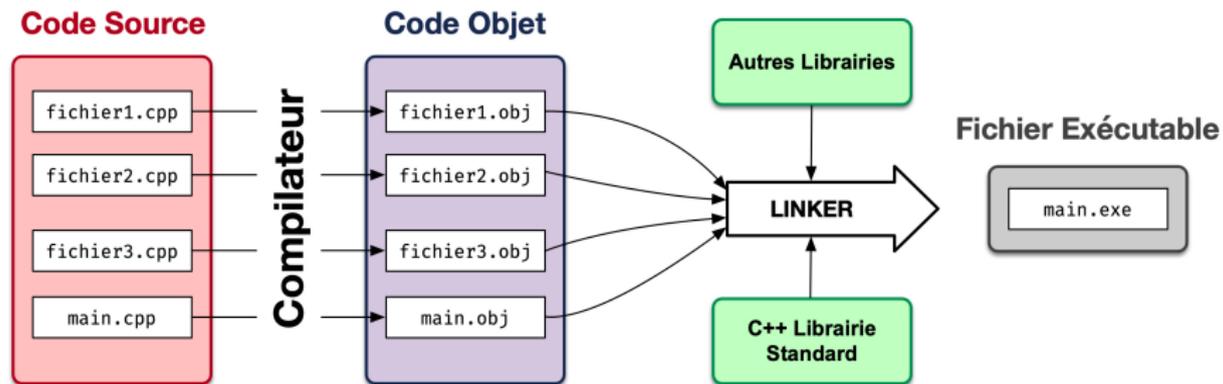
Le code source

- **L'ordinateur est stupide !**
- Vous devez lui dire **exactement** quoi faire, lui donner la recette
- Langage de programmation :
 - Code source
 - Langage de haut niveau
 - Créé pour les humains
- Éditeur :
 - Utilisé pour saisir le code source
 - Fichiers .cpp et .h

Le compilateur

- **Compilateur :**
 - Transforme le code source en fichier binaire
- **Fichiers binaires :**
 - Code de bas niveau
 - Code objet, pour l'ordinateur
- **Linker :**
 - Lie ensemble le code objet et les autres bibliothèques utilisées
 - Crée un programme exécutable

Processus total



Sommaire

1. Pourquoi apprendre le C++ ?
2. Historique
3. Processus de construction du programme
- 4. Structure d'un programme C++**
5. Les commentaires
6. Les directives de préprocesseur
7. La fonction main()
8. L'espace de nom
9. Les flux d'entrée / sortie

Un premier code C++

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int nbFavori;
6
7      cout << "Entrez votre nombre favori :";
8
9      cin >> nbFav;
10
11     cout << "Super ! C'est mon nombre favori aussi !" <<
12     endl;
13     cout << "Non vraiment !" << nbFavori << "est mon nombre
14     favori";
15 }
```

Un premier code C++

>_ Sortie Console :

```
Entrez votre nombre favori : 42  
Super ! C'est mon nombre favori aussi !  
Non vraiment ! 42 est mon nombre favori
```

Les mots-clés

| A - C | D - P | R - Z |
|--------------------------------------|-------------------------------|--|
| <code>alignas</code> (C++11) | <code>decltype</code> (C++11) | <code>constexpr</code> (reflection TS) |
| <code>alignof</code> (C++11) | <code>default</code> (1) | <code>register</code> (2) |
| <code>and</code> | <code>delete</code> (1) | <code>reinterpret_cast</code> |
| <code>and_eq</code> | <code>do</code> | <code>requires</code> (C++20) |
| <code>asm</code> | <code>double</code> | <code>return</code> |
| <code>atomic_cancel</code> (TM TS) | <code>dynamic_cast</code> | <code>short</code> |
| <code>atomic_commit</code> (TM TS) | <code>else</code> | <code>signed</code> |
| <code>atomic_noexcept</code> (TM TS) | <code>enum</code> (1) | <code>sizeof</code> (1) |
| <code>auto</code> (1) (2) (3) (4) | <code>explicit</code> | <code>static</code> |
| <code>bitand</code> | <code>export</code> (1) (3) | <code>static_assert</code> (C++11) |
| <code>bitor</code> | <code>extern</code> (1) | <code>static_cast</code> |
| <code>bool</code> | <code>false</code> | <code>struct</code> (1) |
| <code>break</code> | <code>float</code> | <code>switch</code> |
| <code>case</code> | <code>for</code> (1) | <code>synchronized</code> (TM TS) |
| <code>catch</code> | <code>friend</code> | <code>template</code> |
| <code>char</code> | <code>goto</code> | <code>this</code> (4) |
| <code>char8_t</code> (C++20) | <code>if</code> (2) (4) | <code>thread_local</code> (C++11) |
| <code>char16_t</code> (C++11) | <code>inline</code> (1) | <code>throw</code> |
| <code>char32_t</code> (C++11) | <code>int</code> | <code>throw</code> |
| <code>class</code> (1) | <code>long</code> | <code>true</code> |
| <code>compl</code> | <code>mutable</code> (1) | <code>try</code> |
| <code>concept</code> (C++20) | <code>namespace</code> | <code>typedef</code> |
| <code>const</code> | <code>new</code> | <code>typeid</code> |
| <code>constexpr</code> (C++20) | <code>noexcept</code> (C++11) | <code>typename</code> |
| <code>constexpr</code> (C++11) | <code>not</code> | <code>union</code> |
| <code>constinit</code> (C++20) | <code>not_eq</code> | <code>unsigned</code> |
| <code>const_cast</code> | <code>nullptr</code> (C++11) | <code>using</code> (1) |
| <code>continue</code> | <code>operator</code> (4) | <code>virtual</code> |
| <code>co_await</code> (C++20) | <code>or</code> | <code>void</code> |
| <code>co_return</code> (C++20) | <code>or_eq</code> | <code>volatile</code> |
| <code>co_yield</code> (C++20) | <code>private</code> (3) | <code>wchar_t</code> |
| | <code>protected</code> | <code>while</code> |
| | <code>public</code> | <code>xor</code> |
| | | <code>xor_eq</code> |

- Font partie du vocabulaire du langage.
- Les mots-clés sont réservés !
- C++ contient à peu près **90 mots-clés**
- Javascript \approx 50
- C \approx 32
- Python \approx 33
- Plus un langage a de mots-clés, plus son utilisation est complexe

Les mots-clés

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int nbFavori;
6
7      cout << "Entrez votre nombre favori :";
8
9      cin >> nbFav;
10
11     cout << "Super ! C'est mon nombre favori aussi !" <<
12     endl;
13     cout << "Non vraiment !" << nbFavori << "est mon nombre
14     favori";
15 }
```

Les identifiants

- **Nom donné à une entité** : Variables, fonctions, classes, structures
...
- Nom attribué à une partie du code pour **référencer et utiliser**
- **⚠ Un identifiant doit être différent d'un mot-clé !**

Les identifiants

```
1  #include <iostream>
2  using namespace std;
3
4  main(){
5      int nbFavori;
6
7      cout << "Entrez votre nombre favori :";
8
9      cin >> nbFav;
10
11     cout << "Super ! C'est mon nombre favori aussi !" <<
12     endl ;
13     cout << "Non vraiment !" << nbFavori << "est mon
14     nombre favori";
15 }
```

Les opérateurs

- Opérateurs de flux : `<<` et `>>`
- Opérateurs mathématiques : `+` `x` `-` `/`
- ...

Les opérateurs

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int nbFavori;
6      cout << "Entrez votre nombre favori :";
7
8      cin >> nbFav;
9
10     cout << "Super ! C'est mon nombre favori aussi !"
11         << endl;
12     cout << "Non vraiment !" << nbFavori << "est mon
13         nombre favori";
14 }
```

La ponctuation

- Les points virgules : `;`
- Les accolades : `{ }`
- Les parenthèses : `()`
- Les guillemets : `" "`

La ponctuation

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int nbFavori;
6
7      cout << "Entrez votre nombre favori :";
8
9      cin >> nbFav;
10
11     cout << "Super ! C'est mon nombre favori aussi !" <<
12     endl;
13     cout << "Non vraiment !" << nbFavori << "est mon nombre
14     favori";
15 }
```

Syntaxe

Mots-Clés + Identifiants + Opérateurs +
Ponctuation

Sommaire

1. Pourquoi apprendre le C++ ?
2. Historique
3. Processus de construction du programme
4. Structure d'un programme C++
- 5. Les commentaires**
6. Les directives de préprocesseur
7. La fonction main()
8. L'espace de nom
9. Les flux d'entrée / sortie

Ligne / Paragraphe

- Pour une ligne : `// Texte à commenter`
- Pour un paragraphe : `/* Paragraphe à commenter */`

Exemple

```
1  /*
2  Programme exemple
3  Mon premier programme en C++
4  */
5  #include <iostream>
6  using namespace std;
7
8  int main(){
9      int nbFavori; // Déclaration de la variable nbFavori
10     cout << "Entrez votre nombre favori :";
11     cin >> nbFav;
12     cout << "Super ! C'est mon nombre favori aussi !" <<
13     endl;
14     cout << "Non vraiment !" << nbFavori << "est mon nombre
15     favori";
16     return 0;
17 }
```

Sommaire

1. Pourquoi apprendre le C++ ?
2. Historique
3. Processus de construction du programme
4. Structure d'un programme C++
5. Les commentaires
- 6. Les directives de préprocesseur**
7. La fonction main()
8. L'espace de nom
9. Les flux d'entrée / sortie

Définition

- **Préprocesseur** : Programme qui traite le code avant le compilateur
- Supprime les lignes commencent par un `#` : Les commentaires
- Traite les lignes commencent par un `#` suivies d'une directive préprocesseur :
 - `#include <iostream>`
 - `#include "myfile.h"`
 - `#define PinMoteur 22`
 - `#if` , `#elif` , `#else` , `#endif`
 - ...
- **⚠ Le préprocesseur ne comprend pas le C++**, il prépare juste le code source pour le compilateur.

Sommaire

1. Pourquoi apprendre le C++ ?
2. Historique
3. Processus de construction du programme
4. Structure d'un programme C++
5. Les commentaires
6. Les directives de préprocesseur
7. La fonction main()
8. L'espace de nom
9. Les flux d'entrée / sortie

Définition

- **Tous les programmes C++ doivent avoir exactement 1 fonction `main()`**
- Point de départ de l'exécution du programme.
- `return 0` est la valeur de retour de la fonction `main()`.
 - Si l'exécution s'est bien déroulée → `main()` **retourne 0.**
 - Si un problème durant l'exécution → `main()` **retourne une valeur différente de 0 : Le code d'erreur.**

```
1 int main()  
2 {  
3     // code  
4     return 0;  
5 }
```

Sommaire

1. Pourquoi apprendre le C++ ?
2. Historique
3. Processus de construction du programme
4. Structure d'un programme C++
5. Les commentaires
6. Les directives de préprocesseur
7. La fonction main()
- 8. L'espace de nom**
9. Les flux d'entrée / sortie

Définition

- Pourquoi utilise-t-on `using namespace std` ?
- `std` → espace de nom standard de C++
 - Contient : `cin` , `cout` , `endl`
- Si on met pas `using namespace std` , il faut préciser l'espace de nom à chaque fois :
 - `std::cin` , `std::cout` , `std::endl`
 - `::` est l'opérateur de résolution de portée ou *scope resolution operator*
-  Si `cout` ou `cin` existent dans une autre bibliothèque utilisée, il y aura conflit de nom si namespace pas précisé.

Méthode using namespace

```
1  #include <iostream>
2  using namespace std;
3
4
5  int main(){
6      int nbFavori;
7
8      cout << "Entrez votre nombre favori :";
9
10     cin >> nbFav;
11
12     cout << "Super ! C'est mon nombre favori aussi !" <<
13     endl;
14     cout << "Non vraiment !" << nbFavori << "est mon nombre
15     favori";
16     return 0;
17 }
```

Méthode Scope Resolution Operator

→ En utilisant l'opérateur de résolution de portée :

```
1  #include <iostream>
2
3  int main(){
4      int nbFavori;
5
6      std::cout << "Entrez votre nombre favori :";
7
8      std::cin >> nbFav;
9
10     std::cout << "Super ! C'est mon nombre favori aussi !"
11     << std::endl ;
12     std::cout << "Non vraiment !" << nbFavori << "est mon
13     nombre favori";
14 }
```

Autre méthode

→ Espace de nom précisé juste pour `cin cout endl` :

```
1  #include <iostream>
2  using std::cout;
3  using std::cin;
4  using std::endl;
5
6  int main(){
7      int nbFavori;
8      cout << "Entrez votre nombre favori :";
9      cin >> nbFav;
10     cout << "Super ! C'est mon nombre favori aussi !" <<
11         endl;
12     cout << "Non vraiment !" << nbFavori << "est mon nombre
13         favori";
14     return 0;
15 }
```

Sommaire

1. Pourquoi apprendre le C++ ?
2. Historique
3. Processus de construction du programme
4. Structure d'un programme C++
5. Les commentaires
6. Les directives de préprocesseur
7. La fonction main()
8. L'espace de nom
- 9. Les flux d'entrée / sortie**

Définition

`cin` et `cout` sont des objets représentant les flux.

- `cout` → Flux de sortie
 - Permet d'écrire du texte sur la console
 - S'utilise avec l'opérateur d'insertion : `<<`
- `cin` → Flux d'entrée
 - Permet de capturer du texte saisi au clavier
 - S'utilise avec l'opérateur d'extraction : `>>`

Utiliser cout et «

Insérer des données dans le flux `cout` :

```
1 cout << data;  
2 // data est une variable
```

Enchaînement d'opérateurs :

```
1 cout << "Le contenu de data est " << data;
```

Saut de ligne :

```
1 cout << "Le contenu de data est " << endl;  
2 cout << "Et la suite sur une nouvelle ligne !";
```

Utiliser cin et »

Extraire des données depuis le flux `cin`

```
1 cin >> data;  
2 // data est une variable qui stockera le contenu saisi
```

Enchaînement de captures clavier :

```
1 cin >> data1 >> data2;
```

⚠ Type valeur saisie doit correspondre à type variable qui reçoit :

- Si la saisie est un entier `data` doit être un entier
- Si la saisie est une chaîne de caractère `data` doit être du même type pour ne pas générer d'erreurs.