
🧭 Plan de progression complet : Apprendre C++ en 3 mois

🗓️ Mois 1 : Les bases solides du langage

Semaine 1 : Premiers pas

🎯 Objectif : comprendre la syntaxe et la logique de base du C++.

Contenu :

- Installer ton environnement (VS Code ou Code::Blocks + g++)
- Syntaxe du programme (main(), #include, {})
- Affichage avec cout, saisie avec cin
- Variables et types (int, float, char, string, bool)
- Opérateurs arithmétiques et logiques

Exercices :

- Créer un programme qui additionne deux nombres
- Convertir des températures (°C ↔ °F)
- Calculer le périmètre et la surface d'un rectangle

Semaine 2 : Conditions et boucles

🎯 Objectif : apprendre la logique de décision et de répétition.

Contenu :

- if, else if, else, switch
- Boucles for, while, do...while
- Mots-clés break, continue

Exercices :

- Trouver si un nombre est pair ou impair
- Table de multiplication
- Jeu : devine le nombre mystère (1 à 100)

Semaine 3 : Tableaux, chaînes et fonctions

🎯 Objectif : structurer le code et manipuler des collections.

Contenu :

- Tableaux (int tab[5])
- Boucles pour parcourir les tableaux
- Chaînes de caractères (string) et leurs fonctions (length(), substr(), etc.)
- Fonctions : définition, paramètres, retour
- Récursivité (ex : calcul de factorielle)

Exercices :

- Trouver le plus grand élément d'un tableau
- Inverser une chaîne de caractères
- Calculer la factorielle d'un nombre avec récursion

Semaine 4 : Struct et introduction à la POO

🎯 Objectif : comprendre la structuration des données et l'introduction à l'objet.

Contenu :

- Structures (struct)
- Tableaux de structures
- Notion d'objet, classe et attribut
- Constructeur simple

Exercices :

- Structure "Étudiant" (nom, note, moyenne)
- Classe "Rectangle" avec méthode surface()
- Créer plusieurs objets et afficher leurs propriétés

🗓️ **Mois 2 : Programmation orientée objet + concepts avancés**

Semaine 5 : Classes et encapsulation

🎯 Objectif : maîtriser les classes et la protection des données.

Contenu :

- private, public, protected
- Getters / Setters
- Constructeurs et destructeurs

-

Exercices :

- Classe “CompteBancaire” : dépôt, retrait, solde
- Créer un objet, interagir avec ses méthodes

Semaine 6 : Héritage et polymorphisme

🎯 Objectif : comprendre la réutilisation du code et le comportement dynamique.

Contenu :

- Héritage simple et multiple
- Redéfinition de méthodes (override)
- Fonctions virtuelles et classes abstraites

Exercices :

- Classe “Animal” → “Chien”, “Chat”
- Méthode virtuelle crier() selon le type d’animal
- Classe abstraite “Forme” avec aire() et perimetre()

Semaine 7 : Pointeurs, références et mémoire

🎯 Objectif : comprendre la gestion mémoire.

Contenu :

- Pointeurs (*, &, ->)
- new, delete
- Références (int &r = variable;)
- Passage par pointeur et référence

Exercices :

- Inverser deux variables avec pointeurs
- Créer un tableau dynamique avec new
- Compter les occurrences d’un mot avec pointeurs

Semaine 8 : Exceptions, templates et fichiers

🎯 Objectif : apprendre à rendre ton code générique et sûr.

Contenu :

- try, throw, catch

-
- Templates (template <typename T>)
- Lecture/écriture de fichiers (fstream, ofstream, ifstream)

Exercices :

- Fonction générique pour trouver le max de deux nombres
- Lire un fichier texte et compter les lignes
- Gestion d'erreur avec exceptions

🗓️ **Mois 3 : C++ moderne + projets pratiques**

Semaine 9 : STL et algorithmes

🎯 Objectif : exploiter la puissance de la Standard Template Library.

Contenu :

- vector, map, set, list
- Parcours avec itérateurs
- Fonctions sort, find, count, reverse

Exercices :

- Trier un tableau avec sort
- Créer une liste de notes d'étudiants
- Compter les fréquences des mots dans un texte

Semaine 10 : C++ moderne (C++11 à C++23)

🎯 Objectif : adopter les bonnes pratiques modernes.

Contenu :

- auto, nullptr, constexpr
- Boucles for (auto x : vector)
- unique_ptr, shared_ptr, weak_ptr
- Fonctions lambda [](int x){return x*x;}

Exercices :

- Utiliser unique_ptr pour gérer la mémoire
- Trier un vector avec une fonction lambda

Semaine 11 : Projets pratiques

🎯 Objectif : mettre tout en pratique.

Idées de projets :

- 🔹 Gestionnaire d'étudiants (ajout, suppression, moyenne)
- 🔹 Calculatrice avancée
- 🔹 Jeu de devinette version objet
- 🔹 Mini base de données texte avec fichiers

Semaine 12 : Bonus – Devenir pro

🎯 Objectif : aller au-delà du langage.

🎁 **En bonus : Après 3 mois**

⚙️ **1. Qt Framework (interfaces graphiques)**

Qt est une bibliothèque pour créer **des applications avec interface graphique (GUI)**.
Tu peux créer :

- Des fenêtres, boutons, champs de texte
- Des tableaux dynamiques (QTableWidget)
- Des applications professionnelles (comme VLC, Spotify desktop, etc.)

📚 À apprendre :

- Installer Qt Creator
- Créer une fenêtre avec QMainWindow
- Connecter un bouton à une fonction (connect(signal, slot))
- Gérer les événements (QPushButton, QLabel, QLineEdit, etc.)

🎮 **2. SFML ou SDL (jeux / graphismes)**

Bibliothèques pour créer des **jeux 2D ou applications graphiques**.

📚 Concepts clés :

- Créer une fenêtre (sf::RenderWindow)
- Dessiner des formes, images, texte
- Gérer les entrées clavier/souris
- Boucle de jeu (update / render)

💡 Exemple de projets :

- Jeu Pong
- Snake
- Jeu de tir simple

🧱 3. CMake (compilation avancée)

CMake est un **outil de gestion de projet** pour compiler du code C++ facilement, même sur plusieurs plateformes.

📚 À apprendre :

- Créer un fichier CMakeLists.txt
- Compiler avec cmake . puis make
- Gérer des dépendances (Qt, SFML, etc.)

💡 But : savoir structurer des **gros projets professionnels**.

🧰 4. Git et GitHub

Apprends à versionner ton code et collaborer.

📚 Commandes de base :

- git init
- git add .
- git commit -m "message"
- git push origin main
- git pull

💡 Crée un **portfolio GitHub** avec tes projets C++ !

⚡ 5. Optimisation mémoire et performance

Apprends à écrire du code rapide et efficace :

- Utiliser const & pour éviter des copies inutiles
- Minimiser les allocations new/delete
- Profiling avec valgrind
- Mesurer le temps d'exécution (<chrono>)

💡 Objectif : écrire du **C++ professionnel**, rapide et sans fuite mémoire.
