Installer un IDE

Pour écrire du code en C++, vous avez besoin de trois choses: un éditeur de texte, un compilateur et un débugueur. Un éditeur de texte intelligent qui colore le code est préférable à un simple éditeur comme Bloc-Notes. Vous pouvez obtenir ces trois programmes séparément ou utiliser un IDE (environnement de développement) qui combine les trois. Un IDE rassemble tous les fichiers d'un projet dans une même interface pour faciliter la création d'un programme. Vous pouvez choisir parmi plusieurs environnements de développement pour trouver celui qui vous convient le mieux. Quel que soit l'IDE que vous choisissez, vous pouvez réaliser n'importe quel type de programme. Pour créer un nouveau programme, demandez à l'IDE de préparer un "nouveau projet".

1. **Environnement de développement**

Il existe plusieurs environnements de développement (IDE) gratuits pour programmer en C++. Parmi les plus connus, on trouve Code::Blocks, qui fonctionne sur Windows, Mac et Linux, Visual Studio, disponible uniquement sur Windows, et Xcode, disponible uniquement sur Mac OS X. Les programmeurs expérimentés sous Linux peuvent préférer compiler à la main sans utiliser d'IDE. Tous ces IDE sont performants et vous permettront de suivre ce cours sans problème. Le choix de l'IDE dépend de vos préférences personnelles et de votre système d'exploitation.

Souvent en fonction du langage on peut choisir un IDE plutôt qu'un autre.

2. **Installation Visual Studio**

\\192.168.1.163\public\Softwares\

Copier sur votre pc le répertoire VisualStudioCommunity 4,5Go

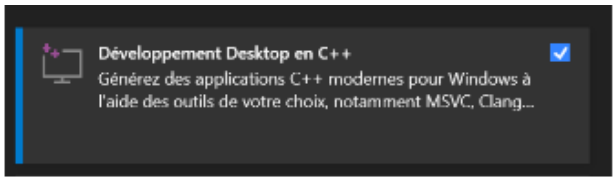
Exécuter VisualStudioSetup.exe

Visual Studio est l'IDE de Microsoft. Il existe une version payante avec des options très évoluées, mais il y a également une version gratuite appelée Visual Studio Community qui permet de programmer en C, C++ et d'autres langages.

Nous utiliserons cette version gratuite dans ce cours. La différence principale avec la version payante est l'absence de l'éditeur de ressources, mais cela n'affectera pas notre utilisation dans ce cours. Pour télécharger Visual Studio Community, rendez-vous sur le site web de Visual Studio Community et sélectionnez "Téléchargez Visual Studio".

https://visualstudio.microsoft.com/vs/community/

Sélectionner


Développement Desktop en C++
Générez des applications C++ modernes pour Windows à l'aide des outils de votre choix, notamment MSVC, Clang...


Charges de travail
Composants individuels
Modules linguistiques
Emplacements d'installation
Web et cloud (4)
Développement web et ASP.NET
Générez des applications web en utilisant ASP.NET Core, ASP.NET, HTML/JavaScript ainsi que des conteneurs pre...
Développement Azure
Kits Azure SDK, outils et projets pour le développement d'applications cloud et la création de ressources à l'aide...
Développement Python
Modification, débogage, développement interactif et contrôle de code source pour Python.
Développement Node.js
Générez des applications réseau scalables via Node.js, un runtime JavaScript piloté par des événements asynchron...
Bureau et mobile (5)
Développement .NET Desktop
Créez des WPF, des Windows Forms et des applications consule à l'aide de C#, de Visual Basic, ainsi que de F# av...
Développement Desktop en C++
Générez des applications C++ modernes pour Windows à l'aide des outils de votre choix, notamment MSVC, Clang...
Développement pour la plateforme Windows universelle
Créez des applications pour la plateforme Windows universelle en C#, VB ou éventuellement C++.
Développement mobile en .NET
Générez des applications multiplateformes pour iOS, Android ou Windows avec Xamarin.
Développement mobile en C++
Générez des applications multiplateformes pour iOS, Android ou Windows en C++.
Jeux (2)
Développement de jeux avec Unity
Créez des jeux 2D et 3D avec Unity, un puissant environnement de développement multiplateforme.
Développement de jeux en C++
Utilisez toute la puissance du C++ pour générer des jeux professionnels basés sur DirectX, Unreal ou Cocos2d.
Autres ensembles d'outils (6)
Stockage et traitement des données
Connectez, développez et testez des solutions de données avec SQL Server, Azure Data Lake ou Hadoop.
Applications de science et analyse des données
Langages et outils permettant de créer des applications de science des données, notamment en Python et F#.
Développement d'extension Visual Studio
Créez des composants additionnels et des extensions pour Visual Studio, notamment de nouvelles commande...
Développement Office/SharePoint
Créez des compléments Office et SharePoint, des solutions SharePoint, ainsi que des compléments VSTO e...
Développement Linux en C++
Créez et déboguez des applications s'exécutant dans un environnement Linux.
Développement .NET multiplateforme
Générez des applications multiplateformes en utilisant .NET, ASP.NET Core, HTML/JavaScript, ainsi que...

Comment écrire et exécuter un programme en C++

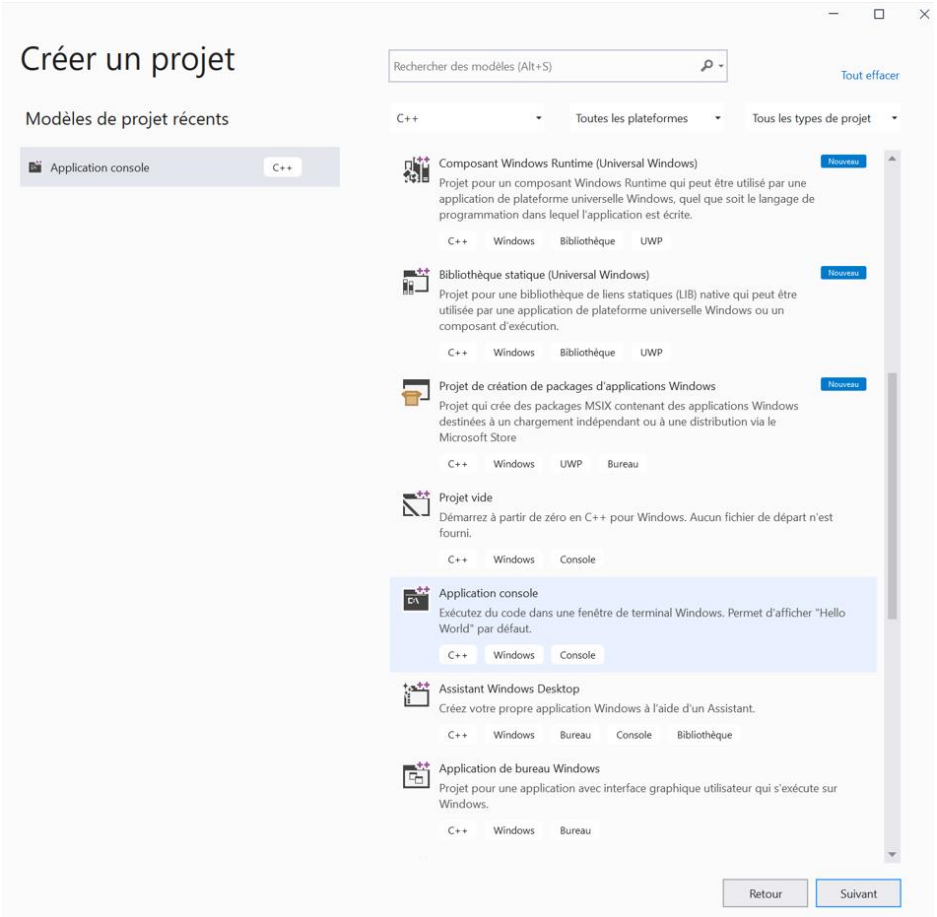
1 . Console ou GUI ?

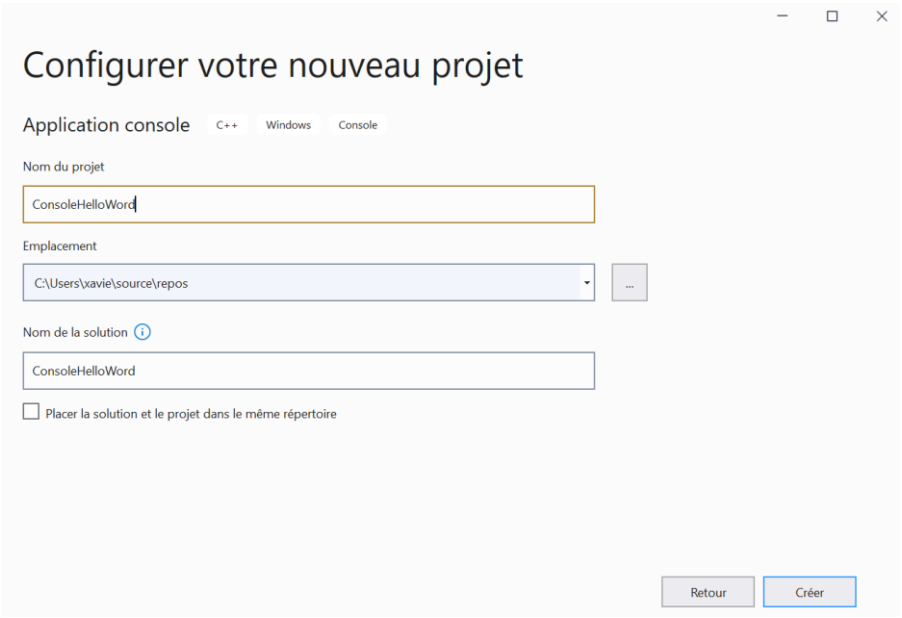
Il existe deux types de programmes:

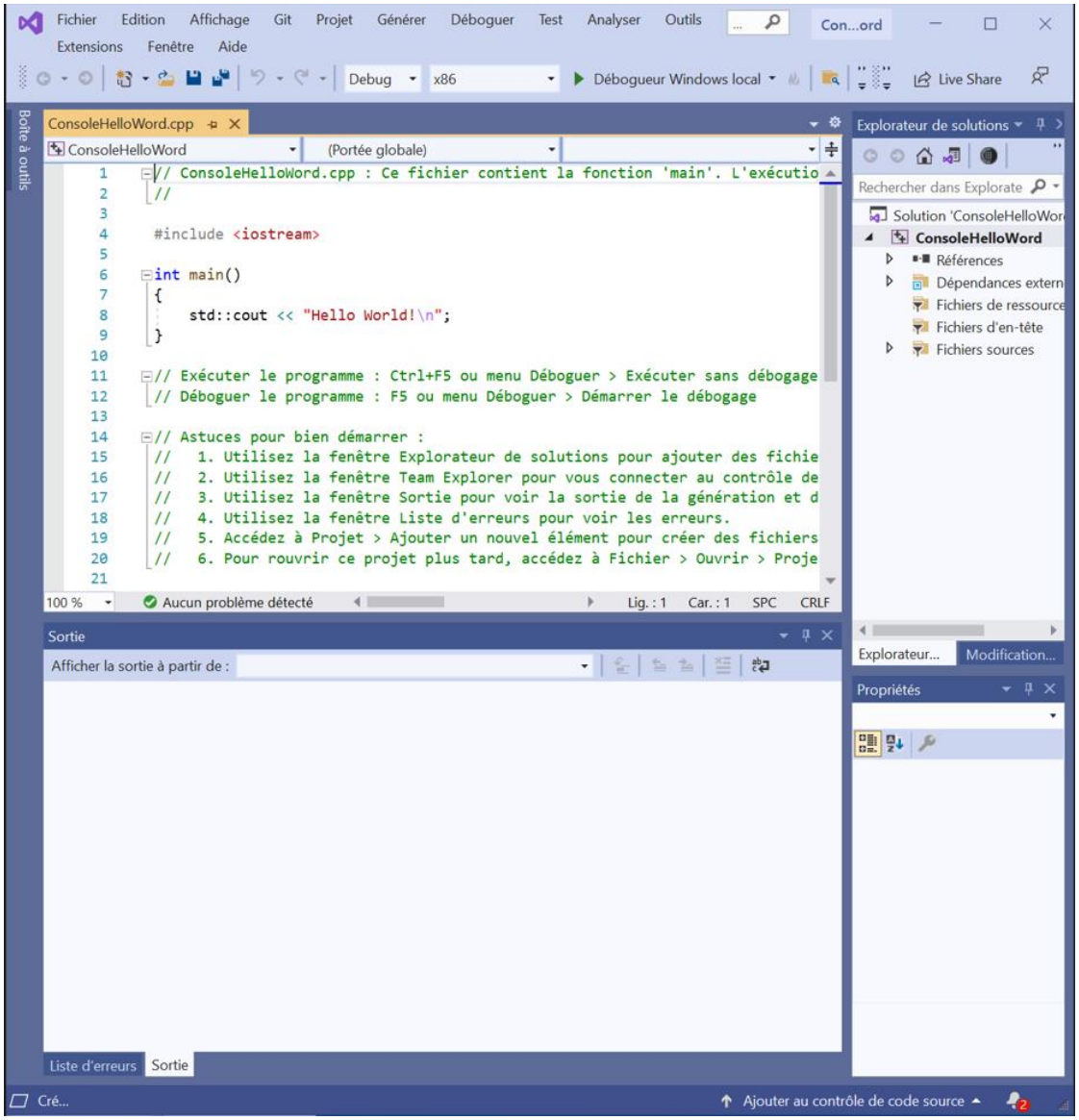
- les programmes graphiques (GUI) = ils affichent des fenêtres interactives, comme Microsoft Word
- les programmes en console = ils affichent du texte en blanc sur fond noir et fonctionnent au clavier

Dans ce cours, vous allez créer des programmes en console, plus simple pour les débutants. Le C++ est plus difficile à maitriser que les autres langages car il demande beaucoup de rigueur. En revanche, ce langage permet de mieux comprendre les interactions avec l'électronique d'un ordinateur.

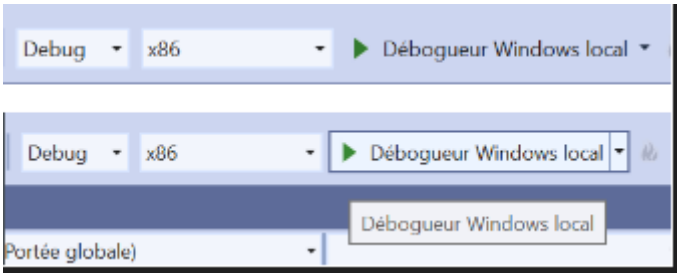
2 . Créer un projet

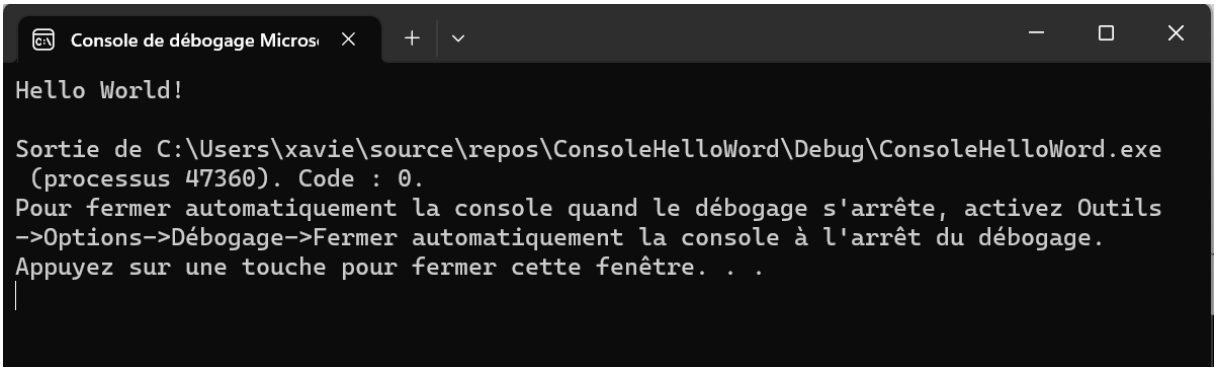



Configurer votre nouveau projet
Application console
C++
Windows
Console
Nom du projet
ConsoleHelloWord
Emplacement
C:\Users\xavie\source\repos
...
Nom de la solution
ConsoleHelloWord
Placer la solution et le projet dans le même répertoire
Retour
Créer


Fichier Edition Affichage Git Projet Générer Déboguer Test Analyser Outils Extensions Fenêtre Aide
Debug
x86
Débogueur Windows local
Live Share
ConsoleHelloWord.cpp
(Portée globale)
// ConsoleHelloWord.cpp : Ce fichier contient la fonction 'main'. L'exécutio
#include <iostream>
int main()
std::cout << "Hello World!\n";
// Exécuter le programme : Ctrl+F5 ou menu Déboguer > Exécuter sans débogage
// Déboguer le programme : F5 ou menu Déboguer > Démarrer le débogage
// Astuces pour bien démarrer :
1. Utilisez la fenêtre Explorateur de solutions pour ajouter des fichie
2. Utilisez la fenêtre Team Explorer pour vous connecter au contrôle de
3. Utilisez la fenêtre Sortie pour voir la sortie de la génération et d
4. Utilisez la fenêtre Liste d'erreurs pour voir les erreurs.
5. Accédez à Projet > Ajouter un nouvel élément pour créer des fichiers
6. Pour rouvrir ce projet plus tard, accédez à Fichier > Ouvrir > Proje
Aucun problème détecté
Lig. : 1
Car. : 1
SPC
CRLF
Explorateur de solutions
ConsoleHelloWord
Références
Fichiers d'en-tête
Fichiers sources
Sortie
Afficher la sortie à partir de :
Liste d'erreurs
Sortie
Propriétés
Ajouter au contrôle de code source

3 . Exécuter le programme en mode debug





Le programme affiche "hello world !", il a été compilé et ensuite exécuté.
Code 0 indique que tout s'est bien passé sans erreur.

Analyse du code Hello World!

```
#include <iostream>


int main()
{
    std::cout << "Hello World!\n";
}
```

Ce code est un programme simple en C++ qui affiche “Hello World!” à l’écran. La première ligne inclut la bibliothèque d’entrée/sortie standard, qui permet d’utiliser std::cout pour afficher du texte à l’écran. La fonction main est la fonction principale du programme, c’est là que commence l’exécution du code. Dans cette fonction, on utilise std::cout pour afficher “Hello World!” suivi d’un saut de ligne à l’écran.

Une directive : #include

La directive #include est une directive de préprocesseur en C++ qui indique au préprocesseur d'inclure le contenu d'un fichier spécifié à l'endroit où la directive apparaît. Vous pouvez organiser les définitions de constantes et de macros en fichiers include (également appelés fichiers d'en-tête) et utiliser des directives #include pour les ajouter à n'importe quel fichier source. Les fichiers Include sont également utiles pour incorporer des déclarations de variables externes et de types de données complexes[1].

Il existe deux formes syntaxiques pour la directive #include:

- #include "path-spec": Le préprocesseur recherche les fichiers Include dans l'ordre suivant: 1) Dans le même répertoire que le fichier qui contient l'instruction #include. 2) Dans les répertoires des fichiers actuellement ouverts, incluez les fichiers, dans l'ordre inverse dans lequel ils ont été ouverts. La recherche commence dans le répertoire du fichier Include parent et continue vers le haut, dans les répertoires de tous les fichiers Include grands-parents. 3) Le long du chemin spécifié par chaque option /I du compilateur. 4) Le long des chemins spécifiés par la variable d'environnement INCLUDE1.
- #include <path-spec>: Le préprocesseur recherche les fichiers Include dans l'ordre suivant:
- 1) Le long du chemin spécifié par chaque option /I du compilateur.
- 2) Lors de la compilation se produit sur la ligne de commande, le long des chemins spécifiés par la variable d'environnement INCLUDE1.

Le préprocesseur arrête de chercher dès qu'il trouve un fichier portant le nom spécifié1. Si vous placez une spécification de chemin complète et non ambiguë pour le fichier include entre guillemets doubles (" "), le préprocesseur recherche uniquement cette spécification de chemin et ignore les répertoires standard1. Si le nom de fichier placé entre guillemets doubles est une spécification de chemin d'accès incomplète, le préprocesseur recherche d'abord le répertoire du fichier parent1.

Une fonction : int main()

La fonction int main() est la fonction principale d'un programme en C++. C'est là que commence l'exécution du code source. Tous les programmes C++ doivent avoir une fonction main. Si vous essayez de compiler un programme C++ sans fonction main, le compilateur générera une erreur[1].

La fonction main renvoie un entier (int) qui représente le code de sortie du programme. Si aucune valeur de retour n'est spécifiée dans main, le compilateur fournit une valeur de retour de zéro[1].

Il existe plusieurs restrictions qui s'appliquent à la fonction main et qui ne s'appliquent à aucune autre fonction C++. La fonction main ne peut pas être surchargée, ne peut pas être déclarée en tant que inline ou static, son adresse ne peut pas être prise et elle ne peut pas être appelée à partir de votre programme[1].

Il existe deux formes courantes pour la fonction main:

- int main(): Cette forme ne prend aucun argument.
- int main(int argc, char* argv[]): Cette forme prend deux arguments, argc et argv, qui permettent de passer des arguments de ligne de commande au programmeargc est un entier qui contient le nombre d'arguments passés au programme, et argv est un tableau de chaînes de caractères qui représentent les arguments passés au programme[2].

En résumé, la fonction int main() est la fonction principale d'un programme en C++ où commence l'exécution du code source. Elle renvoie un entier qui représente le code de sortie du programme et peut prendre des arguments pour passer des arguments de ligne de commande au programme.

Un objet : cout

cout est un objet de la classe ostream en C++ qui est défini dans l'en-tête iostream. Il est utilisé pour afficher la sortie sur le périphérique de sortie standard, c'est-à-dire l'écran. Il est associé au flux de sortie standard C stdout

Le "c" dans cout signifie "caractère" et "out" signifie "sortie". Par conséquent, cout signifie "sortie de caractères".

L'objet cout est utilisé avec l'opérateur d'insertion << pour afficher un flux de caractères. Par exemple, pour afficher une variable entière et une chaîne de caractères, vous pouvez utiliser le code suivant:

```
int var1 = 25;
std::cout << var1;
std::cout << "Some String";
```
Copie

Cela affichera la valeur de var1 suivie de la chaîne de caractères "Some String" à l'écran. Vous pouvez également utiliser plusieurs opérateurs d'insertion << dans une seule instruction pour afficher plusieurs valeurs à la fois.

En résumé, cout est un objet en C++ qui permet d'afficher la sortie sur l'écran en utilisant l'opérateur d'insertion <<.

endl ajoute un retour chariot pour passer à la ligne

Une instruction : using namespace std

using namespace std; est une instruction en C++ qui indique au compilateur d'utiliser l'espace de noms std. Un espace de noms est un conteneur pour un ensemble de déclarations et de définitions, qui permet d'éviter les conflits de noms entre les différentes parties du code.

L’espace de noms std est l’espace de noms standard en C++, qui contient toutes les classes, fonctions et variables de la bibliothèque standard C++. Par exemple, les objets cout et cin, ainsi que les fonctions string et vector, sont tous définis dans l’espace de noms std.

Lorsque vous utilisez l’instruction using namespace std; vous indiquez au compilateur que vous souhaitez utiliser les éléments de l’espace de noms std sans avoir à spécifier leur nom complet. Par exemple, au lieu d’écrire std::cout << "Hello World!";, vous pouvez simplement écrire cout << "Hello World!"; si vous avez utilisé l’instruction using namespace std;.

Il est important de noter que l’utilisation de l’instruction using namespace std; peut parfois causer des conflits de noms si vous utilisez également d’autres espaces de noms ou si vous avez défini vos propres fonctions ou variables avec des noms identiques à ceux de l’espace de noms std. Dans ces cas, il peut être préférable d’éviter d’utiliser cette instruction et de spécifier explicitement l’espace de noms pour chaque élément que vous utilisez.

En résumé, using namespace std; est une instruction en C++ qui indique au compilateur d’utiliser l’espace de noms std, qui contient toutes les classes, fonctions et variables de la bibliothèque standard C++. Cette instruction permet d’utiliser les éléments de cet espace de noms sans avoir à spécifier leur nom complet, mais elle peut parfois causer des conflits de noms.

Ajouter des Commentaires

Les commentaires sont des notes que vous pouvez ajouter à votre code pour expliquer son fonctionnement. Ils sont ignorés par le compilateur et n’apparaissent pas dans le programme généré, mais ils sont utiles pour vous et pour les autres personnes qui lisent votre code. Il existe deux façons d’écrire des commentaires en C++: les commentaires courts, qui commencent par // et sont placés sur une seule ligne, et les commentaires longs, qui s’ouvrent avec /* et se ferment avec */ et peuvent s’étendre sur plusieurs lignes. Les commentaires sont utiles pour expliquer le fonctionnement d’une série d’instructions, mais il n’est pas nécessaire de commenter chaque ligne de code si son fonctionnement est évident.

```
#include <iostream> // Inclut la bibliothèque d'entrée/sortie standard
using namespace std;

int main() // Définit la fonction principale du programme
{
    cout << "Hello World!\n"; // Affiche "Hello World!" suivi d'un saut de ligne à l'écran
}
```

Ajouter un résumé

Il existe deux types de programmes: les programmes graphiques, qui utilisent une interface utilisateur graphique (GUI), et les programmes en console, qui affichent du texte dans une fenêtre de console.

Tout programme doit avoir une fonction main(), qui est le point de départ de l'exécution du code.

La directive cout permet d'afficher du texte dans la console.

Vous pouvez ajouter des commentaires à votre code pour expliquer son fonctionnement en utilisant //Commentaire ou /*Commentaire*/.