

TD 1e — Le même programme en C++ (facultatif)

Guide détaillé, étape par étape

Le TD reprend le programme `bonjour.py` du TD 1c, réécrit en C++ dans `bonjour.cpp`, et l'exécute. Il faut pour cela ajouter à l'éditeur la prise en charge d'un second langage : installer l'extension C/C++, puis un compilateur, que l'extension ne fournit pas. Le programme est compilé, ce qui produit un fichier exécutable, puis cet exécutable est lancé. On compare enfin le nombre de commandes que chacun des deux programmes a demandées, et les fichiers qu'il a laissés sur le disque. Le TD dure une dizaine de minutes une fois le compilateur installé.

Le TD est facultatif : il se fait en séance après les autres TD, si le temps le permet, ou seul ensuite. Sous Windows, l'installation du compilateur nécessite une connexion réseau et prend quelques minutes. VS Code est configuré comme au TD 1c, et son terminal intégré s'ouvre dans l'environnement base d'Anaconda.

Étape	Objectif	Ce qu'on fait
1	donner à l'éditeur un second langage	installer l'extension C/C++
2	disposer d'un compilateur C++	installer un compilateur
3	transformer un source en exécutable, puis le lancer	compiler, puis lancer
4	comparer un programme compilé et un programme interprété	comparer avec le programme Python

Chaque étape commence par un encadré qui la résume. Ce que le TD fait constater est expliqué à la fin du guide, dans « Ce que le TD fait constater » : faire les étapes d'abord, et noter ce qu'on observe, avant de lire l'explication.

1 · Installer l'extension C/C++

À faire : installer l'extension C/C++ de Microsoft ; ouvrir le dossier `cours1\le_hello_cpp\`, puis `bonjour.cpp`.

À obtenir : `bonjour.cpp` s'affiche en couleur, et l'éditeur souligne une faute d'écriture introduite volontairement.

Le dossier du TD

Le dossier ne contient qu'un programme, sans `depart\` ni `travail\` : on travaille directement sur `bonjour.cpp`, qu'on ne modifie qu'à l'étape 4.

```
C:\Users\eleve\Desktop\info01\cours1\le_hello_cpp\  
├─ bonjour.cpp          le programme, en C++  
├─ td_le_hello_cpp.pdf  la feuille du TD  
└─ README.md
```

Dans VS Code, menu File, Open Folder (Fichier, Ouvrir le dossier), puis choisir `Bureau\info01\cours1\le_hello_cpp`. Ouvrir `bonjour.cpp` par un clic dans l'arborescence de gauche.

L'extension

1. Ouvrir le panneau des extensions, `Ctrl + Maj + X`.
2. Taper C/C++ dans la zone de recherche.
3. Choisir « C/C++ », publiée par Microsoft, et cliquer sur Install. Plusieurs extensions portent un nom voisin ; la bonne a pour identifiant `ms-vscode.cpptools`, écrit dans le volet de droite, ligne « Identifier ».

VS Code peut aussi proposer l'extension de lui-même, dans une notification en bas à droite, à l'ouverture d'un fichier `.cpp`.

Vérification : dans `bonjour.cpp`, supprimer le point-virgule à la fin de la ligne `return 0;`. L'éditeur souligne l'endroit en rouge après quelques secondes. Remettre le point-virgule, puis enregistrer (`Ctrl + S`).

À noter : si l'extension permet d'exécuter le programme, ou seulement de l'afficher et d'en vérifier l'écriture.

2 · Installer un compilateur

À faire : vérifier qu'un compilateur C++ est présent ; sous Windows, l'installer dans l'environnement conda.

À obtenir : la commande du compilateur, suivie de `--version`, répond par un numéro de version.

Ouvrir un terminal dans l'éditeur : menu Terminal, New Terminal (Terminal, Nouveau terminal). Il s'ouvre dans le dossier `1e_hello_cpp`.

	Linux, macOS	Windows
Le compilateur	g++, presque toujours déjà installé	aucun, à l'origine
Comment l'obtenir	rien à faire	<code>conda install -c conda-forge "gxx=15.3.0"</code>
Ce qu'on tape ensuite	<code>g++ ...</code>	<code>x86_64-w64-mingw32-g++ ...</code>

Sous Linux et macOS

Taper `g++ --version`, puis Entrée. La réponse commence par un nom et un numéro de version. Sous macOS, si le système propose d'installer les outils de développement en ligne de commande, accepter, puis recommencer.

Sous Windows

Vérifier d'abord que l'invite du terminal commence par `(base)`. Si ce n'est pas le cas, taper `conda activate base`. Puis taper :

```
conda install -c conda-forge "gxx=15.3.0"
```

conda affiche la liste des paquets qu'il va installer, et demande `Proceed ([y]/n)?` : taper `y`, puis Entrée. Le téléchargement dure quelques minutes.

Deux parties de la commande ne doivent pas être modifiées :

- `-c conda-forge` désigne le canal, c'est-à-dire le dépôt de paquets où conda va chercher le compilateur ;
- `"gxx=15.3.0"` fixe la version, avec les guillemets. Sans cette version fixée, conda installe la version 16.2.0, dont la compilation échoue sous Windows (étape 3, « Si la compilation s'arrête sur `crt2.o` »).

Le compilateur installé ne s'appelle pas `g++`. La commande à taper porte le nom complet de la cible, `x86_64-w64-mingw32-g++` :

```
x86_64-w64-mingw32-g++ --version
```

Si conda refuse l'installation (réseau absent, ou droits insuffisants sur l'environnement), le TD se fait chez soi, ou se suit sur le poste de démonstration de l'enseignant. Ne pas installer le paquet `m2w64-toolchain`, proposé par d'anciennes réponses en ligne : sa propre description le déclare obsolète.

Sur un ordinateur personnel qui a déjà MSYS2 et MinGW-w64, installés selon la documentation de VS Code, la commande s'appelle `g++`, comme sous Linux.

Vérification : `x86_64-w64-mingw32-g++ --version` (sous Windows) ou `g++ --version` (sous Linux et macOS) répond par un numéro de version, et non par un message qui dit que la commande est inconnue.

3 · Compiler, puis lancer

À faire : compiler `bonjour.cpp` en un exécutable ; lancer cet exécutable.

À obtenir : le terminal affiche `Bonjour, géomatique !`, et l'arborescence montre un nouveau fichier, `bonjour` (sous Windows, `bonjour.exe`).

Compiler

Dans le terminal, toujours dans `1e_hello_cpp`, taper la commande de compilation, puis Entrée.

	Linux, macOS	Windows
Compiler	<code>g++ bonjour.cpp -o bonjour</code>	<code>x86_64-w64-mingw32-g++ bonjour.cpp -o bonjour.exe</code>
Lancer	<code>./bonjour</code>	<code>.\bonjour.exe</code>

La commande de compilation se lit ainsi : le compilateur, le fichier source à traduire, puis, après `-o` (*output*), le nom du fichier à produire.

La compilation dure quelques secondes, puis l'invite revient.

À noter : ce que la compilation affiche dans le terminal ; ce qui change dans l'arborescence de gauche.

Lancer

Taper la commande de lancement du tableau, puis Entrée. `./` (ou `.\` sous Windows) désigne le dossier courant : il indique au terminal de chercher le programme dans ce dossier.

Vérification : le terminal affiche `Bonjour, géomatique !`.

Si, sous Windows, le `é` s'affiche sous la forme de deux caractères sans rapport, le programme n'est pas en cause : le texte du programme est encodé en UTF-8, et le terminal Windows lit les octets reçus avec une autre table. Taper `chcp 65001`, qui fait passer le terminal en UTF-8, puis relancer le programme.

Si la compilation s'arrête sur `crt2.o`

Avec la version 16.2.0 du compilateur, installée quand la version n'est pas fixée, la compilation se termine ainsi :

```
ld.exe: cannot find crt2.o: No such file or directory
ld.exe: cannot find default-manifest.o: No such file or directory
collect2.exe: error: ld returned 1 exit status
```

La réparation la plus simple est d'installer la version 15.3.0, par la commande de l'étape 2, puis de recompiler. Sans réseau, deux autres réparations conservent la version 16.2.0 ; elles sont écrites pour le terminal `cmd` du TD 1c. La première ajoute une option à chaque compilation :

```
x86_64-w64-mingw32-g++ bonjour.cpp -o bonjour.exe -B "%CONDA_PREFIX%\Library\x86_64-w64-mingw32\sysroot\usr\lib"
```

La seconde répare l'installation une fois pour toutes, par une jonction, qui ne demande pas de droits d'administrateur :

```
mklink /J "%CONDA_PREFIX%\Library\x86_64-w64-mingw32\sysroot\lib" "%CONDA_PREFIX%\Library\x86_64-w64-mingw32\sysroot\usr\lib"
```

Chacune de ces deux commandes s'écrit sur une seule ligne. La cause est expliquée à la fin du guide.

4 · Comparer avec le programme Python

À faire : relever la taille des deux sources et de l'exécutable ; ouvrir l'exécutable dans l'éditeur ; en option, écrire `bonjour.cpp` sur une seule ligne et recompiler.

À obtenir : le tableau de comparaison rempli.

Les tailles

Les commandes suivantes affichent dans le terminal la taille des fichiers, en octets.

	Linux, macOS	Windows (cmd)
Le dossier du TD	<code>ls -l</code>	<code>dir</code>
La source Python du TD 1c	<code>ls -l ../1c_vscode_python/ bonjour.py</code>	<code>dir .. \1c_vscode_python\bonjour.py</code>

Remplir le tableau :

	<code>bonjour.py</code> , TD 1c	<code>bonjour.cpp</code>
Nombre de commandes pour obtenir la phrase		
Ce qui apparaît dans l'arborescence		
Taille du fichier source	121 octets	230 octets
Taille du fichier produit		

Les tailles des sources sont celles des fichiers de l'archive.

Ouvrir l'exécutable dans l'éditeur

Cliquer sur `bonjour` (ou `bonjour.exe`) dans l'arborescence. VS Code prévient que le fichier est binaire, ou qu'il emploie un encodage non pris en charge ; choisir de l'ouvrir quand même. Fermer ensuite l'onglet sans rien enregistrer.

À noter : ce que l'éditeur affiche du fichier produit, comparé à `bonjour.cpp`.

En option : le programme sur une seule ligne

1. Dans `bonjour.cpp`, supprimer la première ligne, le commentaire qui commence par `//` : un commentaire s'étend jusqu'à la fin de la ligne, et masquerait tout ce qui serait écrit après lui.
2. Laisser chacune des deux lignes `#include` seule sur sa ligne.
3. Mettre tout le reste, de `int main()` jusqu'à l'accolade fermante, sur une seule ligne, en supprimant les retours à la ligne.
4. Enregistrer, recompiler, relancer.

À noter : si la compilation réussit, et ce que le programme affiche.

Remettre ensuite `bonjour.cpp` dans son état d'origine, par `Ctrl + Z` répété, ou en le recopiant depuis l'archive de la séance.

Ce que le TD fait constater

Cette section se lit après avoir fait les étapes.

L'extension et le compilateur

L'extension C/C++ apporte la coloration, la vérification de l'écriture et un bouton d'exécution, mais elle n'apporte pas de compilateur : elle analyse le texte C++, sans pouvoir le traduire en programme exécutable. La documentation de VS Code l'indique : « The C/C++ extension doesn't include a C++ compiler or debugger, since VS Code as an editor relies on command-line tools for the development workflow. » Un éditeur de code se configure langage par langage, de la même façon pour Python et pour C++ ; ce qui change d'un langage à l'autre est l'outil qui exécute le programme, l'interpréteur pour Python, le compilateur pour C++.

Python est installé avec Anaconda, alors qu'aucun compilateur C++ ne l'est, et Windows n'en fournit aucun. L'environnement conda sert ici à installer un outil qui n'a rien à voir avec Python. La commande s'appelle `x86_64-w64-mingw32-g++`, nom complet de la cible : l'architecture du processeur (`x86_64`), le système (`w64-mingw32`, Windows 64 bits), puis le compilateur (`g++`).

Deux commandes au lieu d'une

	<code>bonjour.py</code> , TD 1c	<code>bonjour.cpp</code>
Nombre de commandes	une	deux : compiler, puis exécuter
Ce qui apparaît dans l'arborescence	rien	<code>bonjour</code> , un exécutable
Taille du fichier source	121 octets	230 octets
Taille du fichier produit	aucun fichier	environ 20 000 octets

Les deux programmes affichent la même phrase. La compilation n'affiche rien quand elle réussit : elle n'affiche pas de résultat, mais produit un fichier, qui apparaît dans l'arborescence. Le programme Python, lui, n'a rien laissé sur le disque : l'interpréteur lit la source et l'exécute dans la même commande.

La taille de l'exécutable dépend du compilateur et du système : 23 624 octets avec `g++ 11` sous Linux, 19 560 avec `g++ 13`. L'exécutable est de l'ordre de cent fois plus gros que la source, parce qu'il contient le code nécessaire pour démarrer et s'exécuter sans le compilateur. Le fichier Python ne s'exécute pas sans l'interpréteur, qui est installé à part et dont la taille n'est pas comptée ici.

Ouvert dans l'éditeur, l'exécutable est illisible : il contient des instructions pour le processeur, qui ne sont pas du texte. python lui-même est un exécutable de ce type, compilé depuis du C.

Les retours à la ligne en C++

Écrit sur une seule ligne, `bonjour.cpp` se compile et affiche la même phrase. Le compilateur C++ ne tient pas compte des retours à la ligne ni de l'indentation : la fin d'une instruction est marquée par le point-virgule, et un bloc par des accolades. Les lignes `#include` font exception : ce sont des directives, lues avant la compilation, une par ligne. La même opération échoue sur un programme Python qui contient une boucle, où l'indentation et les fins de ligne délimitent les blocs, comme l'a montré le TD 1d.

L'erreur `crt2.o`

Le message de l'étape 3 est écrit par `ld`, l'éditeur de liens : la traduction de `bonjour.cpp` a réussi, et c'est l'assemblage du programme exécutable qui échoue. `crt2.o` est le fichier de démarrage ajouté à tout programme Windows, et exécuté avant `main`. Il est installé, dans `%CONDA_PREFIX%\Library\x86_64-w64-mingw32\sysroot\usr\lib`, mais `gcc 16.2.0` ne le cherche plus dans ce dossier. L'option `-B` lui indique où chercher ; la jonction créée, à l'endroit où il cherche, un accès au dossier réel.

Le défaut vient du paquet `conda-forge` ; le poste et le programme n'y sont pour rien. Il est suivi dans l'issue 229 du dépôt `conda-forge/ctng-compilers-feedstock`, ouverte le 4 septembre 2026 : <https://github.com/conda-forge/ctng-compilers-feedstock/issues/229>. Les versions 13.4.0, 14.4.0 et 15.3.0 n'ont pas ce défaut.