Getting Started With C++

1 Who and When Was C++ Developed

C++ was developed by **Bjarne Stroustrup** at *Bell Labs* starting in 1979. The latest version of C++ is C++23.

What Is C++ Used For Compared to Python

Aspect	C++	Python
Performance	High-performance, compiled	Slower, interpreted
Memory Control	Manual, deterministic	Automatic garbage collection
Use Cases	Systems programming, games, embedded systems, real-time applications	Data science, scripting, automation, web apps
Syntax	Verbose, statically typed	Concise, dynamically typed

2 Installing the development tools

On Windows, we will be programing with C++ on the Linux operating system. We will use Visual Studio Code as the editor.

2.1 Installing Visual Studio Code

- **Windows**: Download the installer from `https://code.visualstudio.com`. Run the setup and ensure options like "Add to PATH" and "Open with Code" are selected.
- **macOS**: Download the `.zip` file from the same site. Drag the app to `/Applications`.

2.2 Running Linux on Windows using Windows Subsystem for Linux (WSL)

Step 1: Enable the "Windows Subsystem for Linux" Feature

1. Click on the Start Menu (the Windows logo in the bottom-left corner).
2. Type in `Turn Windows features on or off` and click on it when it appears.
3. Scroll down the list until you find `Windows Subsystem for Linux`.
4. Check the box next to it.
5. Click OK.
6. Windows might ask you to restart your computer. Go ahead and do that.

Step 2: Install the Ubuntu 24.04 App from the Microsoft Store

7. Click on the Microsoft Store icon on your taskbar (it looks like a shopping bag with the Windows logo). If you don't see it, search for `Microsoft Store` in the Start Menu.

8. In the search bar at the top, type `Ubuntu`.
9. You should see an app named `Ubuntu` (sometimes it might say `Ubuntu on Windows`). Click on it.
10. Look for a button that says `Get` or `Install`. Click on that.
11. Windows will download and install Ubuntu. This might take a few minutes depending on your internet speed.

Step 3: Launch Ubuntu for the First Time
Once the installation is done, you can start using Ubuntu!

12. Click on the Start Menu again.
13. Type `Ubuntu` and click on the `Ubuntu` app that appears.
14. A black window (called a *terminal* or *console*) will open. This is where you'll interact with Ubuntu using text commands.
15. The first time you run it, Ubuntu will take a little while to set things up. It will ask you to create a username and a password for your Ubuntu environment.
16. **Important:** When typing the password, no characters will appear on the screen. That's normal. Just type and press `Enter`.

That's it! You now have Ubuntu 24.04 running on your Windows machine through WSL. You can start exploring the Linux command line and learning programming tools.

A few extra things to keep in mind:

- You can open the Ubuntu terminal anytime by searching for `Ubuntu` in the Start Menu.
- Files created in Ubuntu are generally separate from your Windows files.

For full instructions, see the official guide: `https://learn.microsoft.com/en-us/windows/wsl/install`

2.3 macOS

Install Xcode command line tools from the terminal:

```
xcode-select --install
```

3 Accessing WSL Linux Files from Windows VS Code

You can seamlessly access your WSL Linux files from your Windows-installed VS Code using the **Remote - WSL** extension. This extension essentially allows VS Code running on Windows to interact with your Linux environment as if it were local. Here's how to set it up and use it:

3.1 Install the Remote - WSL Extension

- Open Visual Studio Code on your Windows machine.
- Go to the **Extensions** view (Ctrl+Shift+X or click the Extensions icon in the Activity Bar on the left).
- Search for "`Remote - WSL`" by Microsoft.
- Click **Install**.
- If you plan to use other remote development features in VS Code (like connecting to remote SSH servers or Docker containers), you can instead install the "`Remote Development`" extension pack, which includes the WSL extension.

3.2 Open Your WSL Files in VS Code

There are a few ways to do this once the extension is installed:

3.2.1 From the WSL Terminal

1. Open your WSL terminal (e.g., by searching for "Ubuntu" in the Start Menu).
2. Navigate to the directory you want to open in VS Code using Linux commands (e.g., `cd /home/your_username/your_pro`
3. Type `code .` (note the dot at the end, which signifies the current directory) and press Enter.
4. VS Code will open a new window, and in the bottom-left corner, you'll see a green indicator that says "WSL: [Your Distro Name]" (e.g., "WSL: Ubuntu"). This confirms you are connected to your WSL environment.

3.2.2 From VS Code Directly

1. Open Visual Studio Code on Windows.
2. Press **F1** to open the Command Palette (or Ctrl+Shift+P).
3. Type "`WSL: Connect to WSL`" and press Enter. This will connect to your default WSL distribution.
4. Alternatively, you can type "`WSL: Connect to WSL using Distro...`" if you have multiple WSL distributions and want to choose a specific one.
5. Once connected, go to **File > Open Folder...**
6. A file explorer window will open, but it will be showing the file system *within your WSL environment*. You can navigate to your Linux files here (usually under `/home/your_username/`). Select the folder you want to open and click "OK".

3.2.3 Reopening a Recent WSL Folder

- If you've previously opened a WSL folder in VS Code, you can quickly reopen it via **File ¿ Open Recent** and look for entries with the "[WSL]" suffix.

3.3 Working with WSL Files in VS Code

Once connected to your WSL environment in VS Code:

- **File Explorer:** The Explorer view (Ctrl+Shift+E) will show the files and folders within your WSL Linux file system.
- **Terminal:** When you open a new terminal in VS Code (Terminal > New Terminal, or Ctrl+`), it will be a Linux terminal running within your WSL environment, allowing you to execute Linux commands directly.
- **Editing and Saving:** You can edit your Linux files as you would any other files in VS Code. When you save, the changes are directly saved within your WSL file system.
- **Extensions:** VS Code intelligently manages extensions when working with WSL. Some extensions might run on the Windows side, while others might run within the WSL environment to provide the best experience for your Linux files and tools. You might see a prompt to install certain extensions in WSL if they are recommended for your project type.
- **Debugging:** You can configure debugging for applications running within your WSL environment directly from VS Code (depending on the language and debugging tools you have installed in WSL).

By using the Remote - WSL extension, you can leverage the best of both Windows and Linux for your development workflow!

4 Writing and Running a Hello World Program

We'll create a directory called C++ from the terminal (also command line). Type in the following commands.

```
mkdir C++
ls
cd C++
```

4.1 Source Code (`hello.cpp`)

Now open the C++ folder from VS Code. Create a new file, *hello.cpp*, and cut and past the following code.

```cpp
#include <iostream>
int main() {
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

4.2 Compiling and Running

From the terminal,

```
g++ hello.cpp -o hello
./hello
```

5 Understanding `cout` and `#include <iostream>` in C++

In C++, `cout` (pronounced *see-out*) is used to print output to the screen. It stands for "character output" and is part of the standard input/output stream library.

To use `cout`, you must include the `<iostream>` header at the top of your program:

```
#include <iostream>
```

This tells the compiler to include the Input/Output Stream library, which provides `cout` and other tools such as `cin` (used for input).

- `cout << "Hello, world!"` sends the string to the output stream (your screen).
- `<<` is called the **insertion operator**; it inserts data into the output stream.
- `endl` ends the current line and moves the cursor to the next line.

This is how C++ programs communicate with the user.

The line `std::cout` means `cout` comes from the `std` library. Alteratively, you could insert `using namespace std;` after the `include`